

3/3/4 (Item 4 from file: 351)
DIALOG(R)File 351:Derwent WPI
(c) 2006 Thomson Derwent. All rts. reserv.

013544710 **Image available**
WPI Acc No: 2001-028916/ 200104
XRPX Acc No: N01-022927

**Parallel processing method for detailed analysis of material structure,
involves processing elements for every job unit which is formed as
combination of loops**

Patent Assignee: FUJI XEROX CO LTD (XERF); TAISHO PHARM CO LTD (TAIS);
HONDA MOTOR CO LTD (HOND)

Inventor: INABATA S; KITAMURA K; MIYAKAWA N; OBARA S; TAKASHIMA H; YAMADA S

Number of Countries: 002 Number of Patents: 002

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
JP 2000298658	A	20001024	JP 99105536	A	19990413	200104 B
US 6799151	B1	20040928	US 2000544201	A	20000407	200464

Priority Applications (No Type Date): JP 99105536 A 19990413

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 2000298658	A		35	G06F-015/177	
US 6799151	B1			G06F-017/50	

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-298658

(43)Date of publication of application : 24.10.2000

(51)Int.Cl. G06F 15/177
G06F 15/16
G06F 17/00

(21)Application number : 11-105536

(71)Applicant : FUJI XEROX CO LTD
TAISHO PHARMACEUT CO LTD

(22)Date of filing : 13.04.1999

(72)Inventor : YAMADA SOU
INAHATA SHINJIROU
MIYAGAWA NOBUAKI
TAKASHIMA HAJIME
KITAMURA KAZUYASU
OBARA SHIGERU

(54) PARALLEL PROCESSING METHOD AND PARALLEL PROCESSOR

(57)Abstract:

PROBLEM TO BE SOLVED: To calculate the matrix element of a large scale at high speed through the use of a parallel processing system having an inexpensive communication means and a small memory by permitting plural processor elements to process jobs for every job unit formed by making second and third loops as one group.

SOLUTION: The outermost loop is defined as the loop with respect to the combination (RT) of contractional shells R and T satisfying the relation of $R \leq N$ shell and $R \geq T$. The second loop is defined as the loop with respect to a contractional shell S and the third loop is defined as the loop with respect to a contractional shell U. In this case, values of S and U may be between '1' and R. A function value G (R, S, T and U) is calculated and a part of a prescribed matrix element F is calculated on the inner side of the third loop. R and T form one job unit with the fixed second and third loops as a group and plural processor elements process jobs for every job unit.

```

for(RT=1; RT<=N; RT++)
  for(S=1; S<=R; S++)
    for(U=1; U<=S; U++)
      for(T=1; T<=R; T++)
        if(R<T) continue;
        if(R==T)
          for(S=1; S<=R; S++)
            for(U=1; U<=S; U++)
              G(R,S,T,U)=...;
          F(R,T)=...;
        else
          for(S=1; S<=R; S++)
            for(U=1; U<=S; U++)
              G(R,S,T,U)=...;
          F(R,T)=...;
  F(R,T)=...;

```

LEGAL STATUS

[Date of request for examination]

17.02.2006

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision
of rejection]

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号
特開2000-298658
(P2000-298658A)

(43) 公開日 平成12年10月24日 (2000. 10. 24)

(51) Int.Cl. ⁷	識別記号	F I	ターミナル* (参考)
G 0 6 F 15/177	6 8 1	G 0 6 F 15/177	6 8 1 Z 5 B 0 4 5
15/16	6 1 0	15/16	6 1 0 G 5 B 0 4 9
17/00		15/20	D

審査請求 未請求 請求項の数17 O L (全 35 頁)

(21) 出願番号 特願平11-105536

(22) 出願日 平成11年4月13日 (1999. 4. 13)

(71) 出願人 000005496

富士ゼロックス株式会社
東京都港区赤坂二丁目17番22号

(71) 出願人 000002819

大正製薬株式会社
東京都豊島区高田3丁目24番1号

(72) 発明者 山田 想

神奈川県足柄上郡中井町境430 グリーン
テクなかい富士ゼロックス株式会社内

(74) 代理人 100091546

弁理士 佐藤 正美

最終頁に続く

(54) 【発明の名称】 並列処理方法および並列処理装置

(57) 【要約】

【課題】 安価な通信手段と、小容量のメモリを有する多数のプロセッサエレメントとを用いた並列計算によっても、ホスト計算機とプロセッサエレメントとの間の通信性能に律速されることなく、効率的に行列要素計算を行える。

【解決手段】 例えば、分子軌道計算方法において、フォック行列の全ての要素F (I, J) の計算を、最も外側のループは、 $R \leq N_{shell}$ および $T \leq R$ なる関係を満たす縮約シェルRと縮約シェルTとの組み合わせ

(R T) に関するループとする。2番目は、縮約シェルSに関するループ、3番目は縮約シェルUに関するループとするか、あるいは、2番目は縮約シェルUに関するループ、3番目は縮約シェルSに関するループとする。Sのとりうる値の範囲を1からRの間とし、また、Uのとりうる値の範囲を1からRの間とする。3番目のループの内側で、所定の2電子積分の計算およびその結果を用いた所定のフォック行列要素の一部の計算を行うことにより行う。

```
RT並列アルゴリズム (この発明の実施形態)
for(RT=1; RT<=Nshell*(Nshell+1)/2; RT++){
  for(S=1; S<=R; S++){
    for(U=1; U<=R; U++){
      for(I=b_basis(R); I<=e_basis(R); I++){
        for(J=b_basis(S); J<=e_basis(S); J++){
          for(K=b_basis(T); K<=e_basis(T); K++){
            for(L=b_basis(U); L<=e_basis(U); L++){
              G_ijkl=G(I, J, K, L);
              F[ij]+P[kl]*G_ijkl;
              if(L<=K&&K<I){
                if(J<I) P[kl]+2*P[ji]*G_ijkl;
                else P[kl]+P[ji]*G_ijkl;
              }
              F[ij]-0.5*P[kl]*G_ijkl;
              if(J<I&&K<=J)
                F[kl]-0.5*P[ji]*G_ijkl;
              if(K<I&&J<=K&&L<I)
                F[kl]-0.5*P[ji]*G_ijkl;
            }
          }
        }
      }
    }
  }
}
```

【特許請求の範囲】

【請求項 1】 同じ 1 から N (N は正の整数) の範囲にある 4 つの整数型の変数 R, S, T, U を用いて表わされ、 $G(R, S, T, U) = G(R, S, U, T) = G(S, R, T, U) = G(S, R, U, T) = G(T, U, R, S) = G(T, U, S, R) = G(U, T, R, S) = G(U, T, S, R)$ なる関係を満たす関数 G の関数値 $G(R, S, T, U)$ と； 2 つの変数 T, U を用いて表わされ、 $P(T, U) = P(U, T)$ なる関係を満たす行列 P の要素 $P(T, U)$ と； 係数 A_1 との積 $A_1 \cdot P(T, U) \cdot G(R, S, T, U)$ についての前記範囲の全ての T および U に関する総和 $F_1(R, S)$ と、

前記関数値 $G(R, U, T, S)$ と； 前記行列要素 $P(T, U)$ と； 係数 A_2 との積 $A_2 \cdot P(T, U) \cdot G(R, U, T, S)$ に関する前記範囲の全ての T および U における総和 $F_2(R, S)$ との和 $F(R, S) = F_1(R, S) + F_2(R, S)$ を要素とする行列 F の全要素の計算を、ホスト計算機と、1 つまたは複数のプロセッサエレメントとを有する並列処理装置を用いて行う並列処理方法において、

前記変数 R, S, T, U について、3 重ループを形成し、

前記 3 重ループの最も外側のループは、 $R \leq N$ および $T \leq R$ なる関係を満たす変数 R と変数 T との組み合わせに関するループとし、

前記最も外側のループの内側の 2 番目は前記変数 S に関するループ、前記 2 番目よりも内側の 3 番目は前記変数 U に関するループとするか、あるいは前記 2 番目は変数 U に関するループ、前記 3 番目は変数 S に関するループとし、

前記変数 S のとりうる値の範囲を 1 から R の間とし、前記変数 U のとりうる値の範囲を 1 から R の間とし、前記 3 番目のループの内側で、所定の前記関数値 $G(R, S, T, U)$ の計算およびその計算結果を用いた所定の前記行列要素 F の一部の計算を行うものであって、

前記 2 番目および 3 番目のループをひとまとまりとして 1 つのジョブ単位を形成し、

前記ジョブ単位毎に前記複数のプロセッサエレメントに処理させることを特徴とする並列処理方法。

【請求項 2】 N 個 (N は正の整数) の縮約シェルを用いて表現される分子のエネルギーを計算する分子軌道計算を、ホスト計算機と、1 つまたは複数のプロセッサエレメントとを有する並列処理装置を用いて行う並列処理方法において、

縮約シェル R, S, T, U のそれぞれに含まれる原始シェル r, s, t, u のそれぞれの成分である原始基底関数 i, j, k, l をインデックスとして用いて表わされる 2 電子積分関数 g の関数値 $g(i, j, k, l)$ と；

前記原始基底関数 k をひとつの構成要素とする縮約基底関数 K および前記原始基底関数 l をひとつの構成要素とする縮約基底関数 L とをインデックスとして用いて表わされる密度行列 P の要素 $P(K, L)$ と； 係数 A_1 との積 $A_1 \cdot P(K, L) \cdot g(i, j, k, l)$ の全ての縮約基底関数に関する総和 $f_1(I, J)$ と、

前記 2 電子積分関数の関数値 $g(i, k, j, l)$ と； 前記密度行列 P の要素 $P(K, L)$ と； 係数 A_2 との積 $A_2 \cdot P(K, L) \cdot g(i, k, j, l)$ の全ての縮約基底関数に関する総和 $f_2(I, J)$ との和 $f(I, J) = f_1(I, J) + f_2(I, J)$ の、前記原始基底関数 i, j をひとつの構成要素とする前記縮約基底関数 I, J に含まれる全ての前記原始基底関数に関する和で表わされるフォック行列 F の全ての行列要素 $F(I, J)$ の計算を、

最も外側のループは、 $R \leq N$ および $T \leq R$ なる関係を満たす前記縮約シェル R と T との組み合わせに関するループとし、

前記最も外側のループの内側の 2 番目は前記縮約シェル S に関するループ、前記 2 番目よりも内側の 3 番目は前記縮約シェル U に関するループとするか、あるいは前記 2 番目は前記縮約シェル U に関するループ、前記 3 番目は前記縮約シェル S に関するループとし、

前記縮約シェル S のとりうる値の範囲を 1 から R の間とし、

前記縮約シェル U のとりうる値の範囲を 1 から R の間とし、

前記 3 番目のループの内側で、所定の 2 電子積分の計算およびその結果を用いた所定のフォック行列要素の一部の計算を行うものであって、

前記 2 番目および 3 番目のループをひとまとまりとして 1 つのジョブ単位を形成し、

前記ジョブ単位毎に前記複数のプロセッサ・エレメントに処理を割り当てることを特徴とする並列処理方法。

【請求項 3】 ホスト計算機と、1 つまたは複数のプロセッサエレメントとを備え、請求項 1 または請求項 2 に記載の並列処理方法を実行する並列処理装置であって、前記ホスト計算機は、少なくとも、

前記複数のプロセッサエレメントに対する前記 R と T とが固定されたジョブ単位の割り当て決定と、

前記プロセッサエレメントに対して送信すべき前記行列 P の一部の行列要素の選択と、

前記選択された行列要素の前記プロセッサエレメントに対する送信と、

プロセッサ・エレメントから送信された行列 F の一部の行列要素の受信と、

前記行列 F を用いた前記行列 P の更新と、を行い、

前記プロセッサエレメントは、前記ホスト計算機との間でデータの送受信が可能で、少なくとも、

前記ホスト計算機から送信された前記行列 P の一部の行

列要素の受信と、
 前記 S に関するループの制御と、
 前記 U に関するループの制御と、
 前記関数 $G(R, S, T, U)$ または前記関数 $g(i, j, k, l)$ の計算と、
 前記行列 F の一部の行列要素の計算と、
 前記行列 F の一部の行列要素の前記ホスト計算機に対する送信を行い、かつ、少なくとも前記ホスト計算機から送信された前記行列 P の一部の行列要素と、
 前記ホスト計算機へ送信する前記行列 F の一部の行列要素とを格納するデータ格納手段を備えることを特徴とする並列処理装置。

【請求項 4】請求項 1 に記載の並列処理方法において、前記関数 $G(R, S, T, U)$ に乗じて行列要素 F (R, S) の計算を行うために用いる行列要素 P (T, U) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $G(R, S, T, U)$ に乗じるための行列要素 P (T, U) を除いた全てと、

前記関数 $G(R, S, T, U)$ に乗じて行列要素 F (T, U) の計算を行うために用いる行列要素 P (R, S) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $G(R, S, T, U)$ に乗じるための行列要素 P (R, S) を除いた全てと、

前記関数 $G(R, S, T, U)$ に乗じて行列要素 F (R, U) の計算を行うために用いる行列要素 P (T, S) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $G(R, S, T, U)$ に乗じるための行列要素 P (T, S) を除いた全てと、

前記関数 $G(R, S, T, U)$ に乗じて行列要素 F (S, T) の計算を行うために用いる行列要素 P (R, U) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $G(R, S, T, U)$ に乗じるための行列要素 P (R, U) を除いた全てとを含むデータを、前記変数 S および前記変数 U に関するループのうち、外側に位置するものの開始前に、前記ホスト計算機から前記プロセッサエレメントに転送することを特徴とする並列処理方法。

【請求項 5】請求項 2 に記載の並列処理方法において、前記関数 $g(i, j, k, l)$ に乗じて行列要素 F (I, J) の計算を行うために用いる行列要素 P (K, L) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $g(i, j, k, l)$ に乗じるための行列要素 P (K, L) を除いた全てと、

前記関数 $g(i, j, k, l)$ に乗じて行列要素 F (K, L) の計算を行うために用いる行列要素 P (I, J) のうち、前記プロセッサエレメントで計算されない

ことがあらかじめ判明している、前記関数 $g(i, j, k, l)$ に乗じるための行列要素 P (I, J) を除いた全てと、

前記関数 $g(i, j, k, l)$ に乗じて行列要素 F (I, L) の計算を行うために用いる行列要素 P (K, J) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $g(i, j, k, l)$ に乗じるための行列要素 P (K, J) を除いた全てと、

10 前記関数 $g(i, j, k, l)$ に乗じて行列要素 F (J, K) の計算を行うために用いる行列要素 P (I, L) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $g(i, j, k, l)$ に乗じるための行列要素 P (R, U) または P (I, L) を除いた全てとを含むデータを、前記インデックス S および前記インデックス U に関するループのうち、外側に位置するものの開始前に、前記ホスト計算機から前記プロセッサエレメントに転送することを特徴とする並列処理方法。

20 【請求項 6】請求項 4 に記載の並列処理方法において、前記ホスト計算機から前記プロセッサエレメントへ送信されるデータが、少なくとも、
 前記変数 R を表わすデータと前記変数 T を表わすデータ、または前記変数 R と T との組み合わせを一次的に表わすデータと、

前記関数 $G(R, S, T, U)$ に乗じて行列要素 F (R, S) の計算を行うために用いる行列要素 P (T, U) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $G(R, S, T, U)$ に乗じるための行列要素 P (T, U) を除いた

30 全ての数値データと、
 前記関数 $G(R, S, T, U)$ に乗じて行列要素 F (T, U) の計算を行うために用いる行列要素 P (R, S) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $G(R, S, T, U)$ に乗じるための行列要素 P (R, S) を除いた全ての数値データと、

前記関数 $G(R, S, T, U)$ に乗じて行列要素 F (R, U) の計算を行うために用いる行列要素 P (T, S) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $G(R, S, T, U)$ に乗じるための行列要素 P (T, S) を除いた

40 全ての数値データと、
 前記関数 $G(R, S, T, U)$ に乗じて行列要素 F (R, U) の計算を行うために用いる行列要素 P (T, S) のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $G(R, S, T, U)$ に乗じるための行列要素 P (T, S) を除いた全ての数値データと、

50 前記行列要素 P (T, U) および P (R, U) を特定す

5

る前記変数 U を表わすデータと、
 前記行列要素 $P(R, S)$ および $P(T, S)$ を特定する変数 S を表わすデータと、
 上記変数 U の個数を表わすデータと、
 上記変数 S の個数を表わすデータとから構成されることを特徴とする並列処理方法。

【請求項 7】請求項 5 に記載の並列処理方法において、前記ホスト計算機から前記プロセッサエレメントへ送信されるデータが、少なくとも、
 前記インデックス R を表わすデータと前記インデックス T を表わすデータ、または前記インデックス R と T との組み合わせを一意的に表わすデータと、

前記関数 $g(i, j, k, l)$ に乗じて行列要素 $F(I, J)$ の計算を行うために用いる行列要素 $P(K, L)$ のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $g(i, j, k, l)$ に乗じるための行列要素 $P(K, L)$ を除いた全ての数値データと、

前記関数 $g(i, j, k, l)$ に乗じて行列要素 $F(K, L)$ の計算を行うために用いる行列要素 $P(I, J)$ のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $g(i, j, k, l)$ に乗じるための行列要素 $P(I, J)$ を除いた全ての数値データと、

前記関数 $g(i, j, k, l)$ に乗じて行列要素 $F(I, L)$ の計算を行うために用いる行列要素 $P(K, J)$ のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している前記関数 $g(i, j, k, l)$ に乗じるための行列要素 $P(K, J)$ を除いた全ての数値データと、

前記関数 $g(i, j, k, l)$ に乗じて行列要素 $F(I, L)$ の計算を行うために用いる行列要素 $P(K, J)$ のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数 $g(i, j, k, l)$ に乗じるための行列要素 $P(K, J)$ を除いた全ての数値データと、

前記行列要素 $P(K, L)$ および $P(I, L)$ を特定する、前記インデックス L で代表される縮約基底関数を構成要素とする前記縮約シェル U を表わすデータと、
 前記行列要素 $P(I, J)$ および $P(K, J)$ を特定する、前記インデックス J で代表される縮約基底関数を構成要素とする前記縮約シェル S を表わすデータと、
 前記縮約シェル U の個数を表わすデータと、
 前記縮約シェル S の個数を表わすデータとから構成されることを特徴とする並列処理方法。

【請求項 8】請求項 4 に記載の並列処理方法において、前記プロセッサエレメントで計算された行列要素 $F(R, S)$ 、 $F(T, U)$ 、 $F(R, U)$ 、 $F(S, T)$ の全てを、前記 S および前記 U に関するループのうち、外側に位置するものの終了後に、前記プロセッサエ

6

レメントから前記ホスト計算機に転送することを特徴とする並列処理方法。

【請求項 9】請求項 5 に記載の並列処理方法において、前記プロセッサエレメントで計算された行列要素 $F(I, J)$ 、 $F(K, L)$ 、 $F(I, L)$ 、 $F(J, K)$ の全てを、前記 S および前記 U に関するループのうち、外側に位置するものの終了後に、前記プロセッサエレメントから前記ホスト計算機に転送することを特徴とする並列処理方法。

【請求項 10】請求項 6 または請求項 8 に記載の並列処理方法において、前記プロセッサエレメントから前記ホスト計算機へ送信されるデータが、少なくとも、
 前記プロセッサエレメントで計算された行列要素 $F(R, S)$ 、 $F(T, U)$ 、 $F(R, U)$ 、 $F(T, S)$ から構成されることを特徴とする並列処理方法。

【請求項 11】請求項 7 または請求項 9 に記載の並列処理方法において、

前記プロセッサエレメントから前記ホスト計算機へ送信されるデータが、少なくとも、

前記プロセッサエレメントで計算された行列要素 $F(I, J)$ 、 $F(K, L)$ 、 $F(I, L)$ 、 $F(K, J)$ から構成されることを特徴とする並列処理方法。

【請求項 12】請求項 6 に記載の並列処理方法において、

前記プロセッサエレメントにおける前記変数 S および前記変数 U に関するループ制御は、

前記行列要素 $P(R, S)$ および $P(T, S)$ を特定する変数 S を表わすデータを、その最初から前記変数 S の個数を表わすデータで示される個数まで順次読み取りながら前記変数 S に関するループを制御し、

前記行列要素 $P(T, U)$ および $P(R, U)$ を特定する変数 U を表わすデータを、その最初から前記変数 U の個数を表わすデータで示される個数まで順次読み取りながら前記変数 U に関するループを制御することにより行われることを特徴とする並列処理方法。

【請求項 13】請求項 7 に記載の並列処理方法において、

前記プロセッサエレメントにおける前記 S および前記 U に関するループ制御は、

前記行列要素 $P(I, J)$ および $P(K, J)$ を特定する、インデックス J で代表される縮約基底関数を構成要素とする縮約シェル S を表わすデータを、その最初から前記縮約シェル S の個数を表わすデータで示される個数まで順次読み取りながら前記 S に関するループを制御し、

前記行列要素 $P(K, L)$ および $P(I, L)$ を特定する、インデックス L で代表される縮約基底関数を構成要素とする縮約シェル U を表わすデータを、その最初から前記縮約シェル U の個数を表わすデータで示される個数

7

まで順次読み取りながら前記Uに関するループを制御することにより行われることを特徴とする並列処理方法。

【請求項14】請求項10に記載の並列処理方法において、

行列要素P(R, S)、P(T, U)、P(R, U)、P(T, S)と、行列要素F(R, S)、F(T, U)、F(R, U)、F(T, S)との全てを、前記プロセッサエレメントに備えられたデータ格納手段に格納できない場合に、

前記ホスト計算機は、

前記行列要素P(T, U)、P(R, U)、F(T, U)、F(R, U)の全てを前記データ格納手段に格納可能であれば、

前記行列要素P(T, U)、P(R, U)、F(T, U)、F(R, U)の全てとともに前記データ格納手段に前記行列要素P(R, S)、P(T, S)、F(R, S)、F(T, S)を格納できるように、前記変数Sの範囲を区切って、前記ジョブを複数のサブジョブに分割し、

前記分割された複数のサブジョブを、同一の前記プロセッサエレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素P(T, U)、P(R, U)を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素F(T, U)、F(R, U)を前記プロセッサエレメントから受信し、

前記複数の各サブジョブの開始時に、前記プロセッサエレメントに対して行列要素P(R, S)、P(T, S)を送信し、

前記複数の各サブジョブの終了時に、前記行列要素F(R, S)、F(T, S)を前記プロセッサエレメントから受信するようにし、

前記行列要素P(R, U)、P(T, U)、F(R, U)、F(T, U)の全てを前記データ格納手段に格納することができず、かつ、前記行列要素P(R, S)、P(T, S)、F(R, S)、F(T, S)の全てを、前記データ格納手段に格納可能であれば、

前記行列要素P(R, S)、P(T, S)、F(R, S)、F(T, S)の全てとともに前記データ格納手段に行列要素P(T, U)、P(R, U)、F(T, U)、F(R, U)を格納できるように、前記変数Uの範囲を区切って、前記ジョブを複数のサブジョブに分割し、

前記分割された複数のサブジョブを、同一の前記プロセッサエレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素P(R, S)、P(T, S)を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了

8

時にだけ、前記行列要素F(R, S)、F(T, S)を前記プロセッサエレメントから受信し、

前記複数の各サブジョブの開始時に、前記プロセッサエレメントに対して前記行列要素P(T, U)、P(R, U)を送信し、

前記複数の各サブジョブの終了時に、前記行列要素F(T, U)、F(R, U)を前記プロセッサエレメントから受信することを特徴とする並列処理方法。

【請求項15】請求項11に記載の並列処理方法において、

行列要素P(I, J)、P(K, L)、P(I, L)、P(K, J)と、行列要素F(I, J)、F(K, L)、F(I, L)、F(K, J)の全てをプロセッサエレメントに備えられたデータ格納手段に格納できない場合に、

前記ホスト計算機は、

前記行列要素P(K, L)、P(I, L)、F(K, L)、F(I, L)の全てを前記データ格納手段に格納可能であれば、

前記行列要素P(K, L)、P(I, L)、F(K, L)、F(I, L)の全てとともに前記データ格納手段に前記行列要素P(I, J)、P(K, J)、F(I, J)、F(K, J)を格納できるように、前記縮約シエルSの範囲を区切って、前記ジョブを複数のサブジョブに分割し、

前記分割された複数のサブジョブを同一の前記プロセッサエレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素P(K, L)、P(I, L)を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素F(K, L)、F(I, L)を前記プロセッサエレメントから受信し、

前記複数の各サブジョブの開始時に、前記プロセッサエレメントに対して前記行列要素P(I, J)、P(K, J)を送信し、

前記複数の各サブジョブの終了時に、前記行列要素F(I, J)、F(K, J)を前記プロセッサエレメントから受信するようにし、

前記行列要素P(I, L)、P(K, L)、F(I, L)、F(K, L)の全てを前記データ格納手段に格納することができず、かつ、前記行列要素P(I, J)、P(K, J)、F(I, J)、F(K, J)の全てを前記データ格納手段に格納可能であれば、

前記行列要素P(I, J)、P(K, J)、F(I, J)、F(K, J)の全てとともに前記データ格納手段に行列要素P(K, L)、P(I, L)、F(K, L)、F(I, L)を格納できるように、前記縮約シエルUの範囲を区切って、前記ジョブを複数のサブジョブに分割し、

10

20

30

40

50

前記分割された複数のサブジョブを、同一の前記プロセッサ・エレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサ・エレメントに対して前記行列要素 $P(I, J)$ 、 $P(K, J)$ を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素 $F(I, J)$ 、 $F(K, J)$ を前記プロセッサ・エレメントから受信し、

前記複数の各サブジョブの開始時に、前記プロセッサ・エレメントに対して前記行列要素 $P(K, L)$ 、 $P(I, L)$ を送信し、

前記複数の各サブジョブの終了時に、前記行列要素 $F(K, L)$ 、 $F(I, L)$ を前記プロセッサ・エレメントから受信することを特徴とする並列処理方法。

【請求項 16】請求項 10 に記載の並列処理方法において、

行列要素 $P(R, S)$ 、 $P(T, U)$ 、 $P(R, U)$ 、 $P(T, S)$ と、行列要素 $F(R, S)$ 、 $F(T, U)$ 、 $F(R, U)$ 、 $F(T, S)$ の全てを、前記プロセッサ・エレメントに備えられたデータ格納手段に格納できない場合に、

前記ホスト計算機は、

前記行列要素 $P(R, S)$ と前記行列要素 $P(T, S)$ の個数の和が、前記行列要素 $P(R, U)$ と前記行列要素 $P(T, U)$ の個数の和よりも大きく、かつ、前記行列要素 $P(T, U)$ 、 $P(R, U)$ 、 $F(T, U)$ 、 $F(R, U)$ の全てを、前記データ格納手段に格納可能でなければ、

前記行列要素 $P(T, U)$ 、 $P(R, U)$ 、 $F(T, U)$ 、 $F(R, U)$ が前記データ格納手段に格納可能となるように、前記変数 U の範囲を等分割し、

前記等分割された前記行列要素 $P(T, U)$ 、 $P(R, U)$ 、 $F(T, U)$ 、 $F(R, U)$ とともに前記データ格納手段に前記行列要素 $P(R, S)$ 、 $P(T, S)$ 、 $F(R, S)$ 、 $F(T, S)$ を格納できるように、前記変数 S の範囲を区切って、前記ジョブを複数のサブジョブに分割し、

前記分割された複数のサブジョブを、同一の前記プロセッサ・エレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサ・エレメントに対して前記行列要素 $P(T, U)$ 、 $P(R, U)$ を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素 $F(T, U)$ 、 $F(R, U)$ を前記プロセッサ・エレメントから受信し、

前記複数の各サブジョブの開始時に、前記プロセッサ・エレメントに対して前記行列要素 $P(R, S)$ 、 $P(T, S)$ を送信し、

前記複数の各サブジョブの終了時に、前記行列要素 $F(R, S)$ 、 $F(T, S)$ を前記プロセッサ・エレメント

から受信するようにし、

前記行列要素 $P(R, S)$ と前記行列要素 $P(T, S)$ の個数の和が、前記行列要素 $P(R, U)$ と前記行列要素 $P(T, U)$ の個数の和よりも小さいか、または等しく、かつ、前記行列要素 $P(R, S)$ 、 $P(T, S)$ 、 $F(R, S)$ 、 $F(T, S)$ の全てを前記データ格納手段に格納可能でなければ、

前記行列要素 $P(R, S)$ 、 $P(T, S)$ 、 $F(R, S)$ 、 $F(T, S)$ が前記データ格納手段に格納可能となるように、前記変数 S の範囲を等分割し、

前記等分割された前記行列要素 $P(R, S)$ 、 $P(T, S)$ 、 $F(R, S)$ 、 $F(T, S)$ とともに前記データ格納手段に前記行列要素 $P(T, U)$ 、 $P(R, U)$ 、 $F(T, U)$ 、 $F(R, U)$ を格納できるように、前記変数 U の範囲を区切って、前記ジョブを複数のサブジョブに分割し、

前記分割された複数のサブジョブを、同一の前記プロセッサ・エレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサ・エレメントに対して前記行列要素 $P(R, S)$ 、 $P(T, S)$ を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素 $F(R, S)$ 、 $F(T, S)$ を前記プロセッサ・エレメントから受信し、

前記複数の各サブジョブの開始時に、前記プロセッサ・エレメントに対して前記行列要素 $P(T, U)$ 、 $P(R, U)$ を送信し、

前記複数の各サブジョブの終了時に、前記行列要素 $F(T, U)$ 、 $F(R, U)$ を前記プロセッサ・エレメントから受信する、

ことを特徴とする並列処理方法。

【請求項 17】請求項 11 に記載の並列処理方法において、

行列要素 $P(I, J)$ 、 $P(K, L)$ 、 $P(I, L)$ 、 $P(K, J)$ と、行列要素 $F(I, J)$ 、 $F(K, L)$ 、 $F(I, L)$ 、 $F(K, J)$ の全てを、前記プロセッサ・エレメントに備えられたデータ格納手段に格納できない場合に、

前記ホスト計算機は、

前記行列要素 $P(I, J)$ と前記行列要素 $P(K, J)$ の個数の和が、前記行列要素 $P(I, L)$ と前記行列要素 $P(K, L)$ の個数の和よりも大きく、かつ、前記行列要素 $P(K, L)$ 、 $P(I, L)$ 、 $F(K, L)$ 、 $F(I, L)$ の全てを前記データ格納手段に格納可能でなければ、

行列要素 $P(K, L)$ 、 $P(I, L)$ 、 $F(K, L)$ 、 $F(I, L)$ を前記データ格納手段に格納可能となるように縮約シェル U の範囲を等分割し、

前記等分割された行列要素 $P(K, L)$ 、 $P(I, L)$ 、 $F(K, L)$ 、 $F(I, L)$ とともに前記データ

格納手段に行列要素 $P(I, J)$ 、 $P(K, J)$ 、 $F(I, J)$ 、 $F(K, J)$ を格納できるように縮約シェル S の範囲を区切ってジョブを複数のサブジョブに分割し、

前記のように区切られた複数のサブジョブを同一の前記プロセッサエレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ前記プロセッサ・エレメントに対して行列要素 $P(K, L)$ 、 $P(I, L)$ を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ行列要素 $F(K, L)$ 、 $F(I, L)$ を前記プロセッサ・エレメントから受信し、

前記複数の各サブジョブの開始時に前記プロセッサエレメントに対して行列要素 $P(I, J)$ 、 $P(K, J)$ を送信し、

前記複数の各サブジョブの終了時に行列要素 $F(I, J)$ 、 $F(K, J)$ を前記プロセッサエレメントから受信するようにし、

行列要素 $P(I, J)$ と前記行列要素 $P(K, J)$ の個数の和が行列要素 $P(I, L)$ と前記行列要素 $P(K, L)$ の個数の和よりも小さいか、または等しく、かつ、前記行列要素 $P(I, J)$ 、 $P(K, J)$ 、 $F(I, J)$ 、 $F(K, J)$ の全てを前記データ格納手段に格納可能でなければ、

前記行列要素 $P(I, J)$ 、 $P(K, J)$ 、 $F(I, J)$ 、 $F(K, J)$ を前記データ格納手段に格納可能となるように、前記縮約シェル S の範囲を等分割し、

前記等分割された前記行列要素 $P(I, J)$ 、 $P(K, J)$ 、 $F(I, J)$ 、 $F(K, J)$ とともに前記データ格納手段に前記行列要素 $P(K, L)$ 、 $P(I, L)$ 、 $F(K, L)$ 、 $F(I, L)$ を格納できるように、前記縮約シェル U の範囲を区切って、前記ジョブを複数のサブジョブに分割し、

前記分割された複数のサブジョブを、同一の前記プロセッサエレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素 $P(I, J)$ 、 $P(K, J)$ を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素 $F(I, J)$ 、 $F(K, J)$ を前記プロセッサエレメントから受信し、

前記複数の各サブジョブの開始時に、前記プロセッサエレメントに対して行列要素 $P(K, L)$ 、 $P(I, L)$ を送信し、

前記複数の各サブジョブの終了時に、前記行列要素 $F(K, L)$ 、 $F(I, L)$ を前記プロセッサエレメントから受信することを特徴とする並列処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 この発明は、大規模で、特定

の対称性を有する行列要素計算、特に、非経験的分子軌道法を用いた分子シミュレーションにおいてフォック行列要素計算を高速に処理するために用いて好適な、並列処理方法および並列処理装置に関する。

【0002】

【従来の技術】 化学の分野において、分子の状態や挙動を数値的に解析する手法として、分子軌道法、分子動力学法、モンテカルロ法などがある。その中でも、非経験的分子軌道計算法は、第一原理に基づいた量子力学的計算で、分子中の電子の挙動を記述することを目的としている。そのため、この手法は、分子シミュレーションの基盤として位置づけられ、物質構造や化学反応の詳細な解析に用いられている工業的に重要な手法である。

【0003】 非経験的分子軌道計算法では、分子を構成する原子の原子核と軌道電子との距離の 2 乗に経験的な定数を乗じたものを指数とする指数関数の逆数あるいはそれらの線形結合を基底関数とし、この基底関数を 1 つの原子に対して複数個用意する。これらの基底関数の線形結合で、分子内の軌道電子の波動関数、すなわち分子軌道を記述する。

【0004】 分子軌道における基底関数の線形結合係数を決めることが、非経験的分子軌道計算法での主要な処理であるが、その計算には、基底関数の数の 4 乗に比例した計算量と記憶容量を必要とする。そのため、非経験的分子軌道計算法は、現状では、100 原子程度の規模の分子系に適用されているに過ぎない。生命現象／化学現象の分子論的解明を、より現実的なものとするためには、数 1000 原子規模の分子系への適用も視野に入れた、非経験的分子軌道計算専用の計算システムの開発が必須である。

【0005】 [非経験的分子起動法の概要] 非経験的分子軌道計算法では、分子の状態 Ψ を、分子中の電子の空間的な軌道に相当する電子軌道関数 ϕ_μ を用いて記述する。ここで μ は複数ある分子軌道の μ 番目という意味の添え字である。分子軌道 ϕ_μ は、原子軌道 χ_I の線形結合で、図 23 の (数式 1) のように近似的に表わされる。

【0006】 ここで、(数式 1) において、 I は複数ある原子軌道の I 番目という意味の添え字である。なお、原子軌道は基底関数とも呼ばれることがある。この明細書中では、以降、原子軌道のことを基底関数と呼ぶ。また、数式 1 に現れる $C_{I\mu}$ は、線形結合係数である。数式 1 における I に関する総和は、計算の対象とする分子を構成する全ての基底関数に関するものである。

【0007】 さて、量子力学的に分子軌道を記述するためには、良く知られるパウリの排他律を、分子内の電子の状態が満たさなければならない。電子のスピンを考慮に入れて、パウリの排他律を満たすように、 $2n$ 電子系の分子の状態 Ψ を記述する表式として、図 23 の (数式 2) のようなスレーター行列式が用いられる。ここで、

(数式2)において、 $\alpha(x)$ および $\beta(x)$ 、 x 番目の電子のスピンが、それぞれ、上向きおよび下向きの状態を表わしている。

【0008】2n電子系に対するハミルトニアンHは、1電子部分 H_1 と2電子部分 H_2 との和という形式で、図23の(数式3)から(数式5)のように書き表される。

【0009】図23の(数式4)において、右辺の(・・・)内の第1項は、電子pの運動エネルギー、第2項はp番目の電子とA番目の原子核との相互作用である。

(数式4)において、 Σ_p (この明細書で Σ_i は、 i についての総和を取ることを表すものとする。以下、同じ)は全電子に関する総和、 Σ_A は全原子核に関する総和、 Z_A は原子核Aの電荷、 r_{pA} は電子pと原子核Aとの距離である。

【0010】また、(数式5)は、電子pと電子qとの間の相互作用を表わしており、 $\Sigma_p \Sigma_q (>p)$ は2個の電子の組み合わせに関する総和、 r_{pq} は電子p、q間の距離である。

【0011】上記のハミルトニアンHと、(数式2)のスレーター行列式とを用いると、分子エネルギーの期待値 ϵ が、図24の(数式6)～(数式9)のように表わされる。

【0012】(数式6)において、 Σ_μ および Σ_ν は、 n 個(n は正の整数)ある分子軌道に関する総和である。(数式7)は「コア積分」と呼ばれ、代表として番号1の電子について書かれている。また、(数式8)および(数式9)は、それぞれ「クーロン積分」および「交換積分」と呼ばれ、代表として電子1および電子2について書かれている。

【0013】(数式6)を基底関数を用いて書き直すと、図25に示す(数式10)～(数式13)に示すようなものになる。(数式13)で表わされる積分を、「2電子間反発積分」あるいは省略して「2電子積分」と呼ぶ。

【0014】(数式10)で表わされる分子エネルギーの期待値 ϵ は、 $C_{I\mu}$ という未知数を含んでおり、このままでは数値が得られない。 $C_{I\mu}$ は、(数式1)における線形結合定数であり、 μ は1から n (分子軌道の数)の整数、 I は1から N (N が基底関数の数であり、正の整数)の整数である。以下では、 $C_{I\mu}$ を要素とする $N \times n$ 行列 C を係数行列と呼ぶ。

【0015】期待値 ϵ が最小となるように係数行列を決定し、基底状態の波動関数 Ψ を求める手法の1つとして、ハートリー・フォック・ローサーンの変分法(以下、HFR法と略称する)が用いられる。導出過程は省略し、HFR法の結果として得られる式を、図26の(数式14)～(数式18)に示す。

【0016】 F_{IJ} はフォック行列要素、 P_{KL} は密度行列要素と、それぞれ呼ばれる。以下の説明では、これらを

$F(I, J)$ 、 $P(K, L)$ のように表記する場合がある。これらは、1から N の値をとる各 I, J, K, L に対して数値を持っており、それぞれ $N \times N$ 行列の形で表わされる。

【0017】(数式14)を解くことにより、係数行列が求まる。(数式14)は、1から n の間の全ての μ 、および1から N の間の全ての I に対して存在するので、 $n \times N$ 本の連立方程式になっている。

【0018】(数式14)を解いて得られた係数行列 C の計算には、密度行列 P が用いられている。密度行列 P は、(数式18)に示すように係数行列 C から計算される。そのため、具体的な計算手順としては、まず、適当に係数行列 C を与えておき、それを用いて計算した密度行列 P を使って、(数式15)でフォック行列 F を計算し、(数式14)の連立方程式を解いて新たな係数行列 C を得る。密度行列 P の元となる C と、結果として得られる C との間の差が十分小さく、すなわち自己無撞着になるまで、上記の計算を繰り返し行う。この反復計算を自己無撞着計算(以下、SCF計算と称する)と呼ぶ。

【0019】実際の計算で最も時間を要するのは、(数式15)のフォック行列要素 F_{IJ} の計算である。これは、全ての I, J に対して、この(数式15)を計算しなければならないこと、および各 I, J の組み合わせに対して、密度行列要素 P_{KL} の K, L に関する和を計算しなければならないことに起因する。

【0020】SCF計算の手法には2通りある。1つはディスクストレージSCF法と呼ばれる手法で、1回目のSCF計算の際に得た2電子積分の値を全てディスクに保存しておき、2回目以降は必要な2電子積分をディスクから取り出して用いる手法である。もう1つはダイレクトSCF法と呼ばれる手法で、SCF計算の度に2電子積分の計算をやり直す手法である。

【0021】現在では、ディスク容量の制限やアクセス時間の大きさなどから、後者のダイレクトSCF法を用いるのが主流である。このダイレクトSCF法による分子軌道計算においては、SCF計算の1回あたりに、 N^4 にほぼ比例する個数の2電子積分の計算を行わなければならないため、2電子積分計算を高速に行うことが分子軌道計算を高速化することに直結する。

【0022】2電子積分 $G(I, J, K, L)$ 、密度行列 $P(K, L)$ 、およびフォック行列 $F(I, J)$ の対称性に関して、ここで言及しておく。

【0023】2電子積分は、(数式13)から明らかにように、図26の(数式19)に示すような対称性を有している。したがって、(数式19)の内の1つに関して数値を得ることができれば、他の7つについても数値が得られたことになる。

【0024】また、図26の(数式18)から、 $P(K, L) = P(L, K)$

であることがわかり、図26の(数式15)および図2

5の(数式11)から、
 $F(I, J) = F(J, I)$
 であることがわかる。

【0025】[縮約基底関数と原始基底関数] 非経験的分子軌道法では、図27の(数式20)に示すような基底関数が一般的に用いられる。この(数式20)において、 r, n, R はベクトルであり、添え字 x, y, z の付いたものがその成分である。 r は電子の座標、 n は電子の角運動量、 R は原子核の座標である。

【0026】 $n_x + n_y + n_z = \lambda$ は、角運動量の大きさであり、軌道量子数とも呼ばれる。この軌道量子数 λ が、0の場合にその軌道を s 軌道、1の場合にその軌道を p 軌道、2の場合にその軌道を d 軌道などと呼ぶ。

【0027】 ζ_m は軌道指数であり、軌道の空間的な広がり具合を示す。軌道指数の異なる複数の軌道の線形結合で1つの基底関数を表わす場合があり、そのようにして表わした基底関数を縮約基底関数と呼び、線形結合係数 d_m を縮約係数と呼ぶ。これに対して、線形結合される前の、図27の(数式21)の形の関数 ψ を原始基底関数と呼ぶ。

【0028】縮約基底関数 χ は、 I, J, K, L のように大文字で番号付けをし、また、原始基底関数 ψ は、 i, j, k, l のように小文字で番号付けするのが慣例であり、本明細書中でもこれに従う。

【0029】[縮約シェルと原始シェル] 軌道量子数が1の場合の縮約基底関数には、 $n = (1, 0, 0)$ の場合、 $n = (0, 1, 0)$ の場合、 $n = (0, 0, 1)$ の場合の3通りが存在する。同様に、軌道量子数が2の場合には6通り(あるいは、基底関数の構成の仕方によっては5通り)の縮約基底関数が存在する。

【0030】(数式20)のうちの図27の(数式22)で示す部分が共通な、これら複数の縮約基底関数の集合を、縮約シェルと呼ぶ。 p 軌道の縮約シェルは3つの縮約基底関数で構成され、また、 d 軌道の縮約シェルは6つ(または5つ)の縮約基底関数で構成される。 s 軌道の場合にも、便宜上1つの縮約基底関数の集合を縮約シェルと呼ぶ。

【0031】(数式21)のうちの $\exp[-\zeta(r-R)^2]$ の部分が共通な、原始基底関数の集合を、同様に原始シェルと呼ぶ。縮約シェルは、 R, S, T, U のように大文字で番号付けをし、原始シェルは、 r, s, t, u のように小文字で番号付けするのが慣例であり、本明細書中でもこれに従う。

【0032】分子軌道計算の実施に際しては、計算の対象とする分子を構成する原子毎に軌道量子数の異なる複数の縮約シェルを用意し、それら全ての集合を基底関数のセットとして用いる。原子核座標 R と軌道量子数 λ との組み合わせ (R, λ) で、1つの縮約シェルを表わすことができる。

【0033】[2電子積分の表式] 縮約基底関数で表わ

される2電子積分 $G(I, J, K, L)$ は、原始基底関数を用いると、図27の(数式23)のように表わされる。ここで、 $g(i, j, k, l)$ は、図27の(数式24)のように表すことができる。

【0034】 $G(I, J, K, L)$ を、縮約基底関数で表現した2電子積分と呼び、 $g(i, j, k, l)$ を、原始基底関数で表現した2電子積分と呼ぶが、以降の説明では、どちらも単に2電子積分と呼ぶ場合がある。 $g(i, j, k, l)$ も、図27の(数式25)で示すような対称性を有している。

【0035】さて、原始基底関数 ψ は、その角運動量 n 、軌道指数 ζ 、原子核座標 R の組み合わせで、一意的に示すことができる。 i, j, k, l 番目の原始基底関数が、図19に示す表1のような角運動量、軌道指数、原子核座標を有するものと仮定する。

【0036】説明の煩雑さを避けるために、以下の説明では、原始基底関数の番号 i, j, k, l の代わりに、それぞれの角運動量 a, b, c, d を用いて2電子積分を $[ab, cd]$ のように表わすことにする。

【0037】上記のように用意された基底関数セットを用いて2電子積分を計算する効率的な手法を、文献1 (S. Obara and A. Saika, JCP, vol. 84, no. 7, p. 3964, 1986) に従って説明する。

【0038】まず、 a, b, c, d が全て s 軌道、すなわち $a = 0_a = (0, 0, 0)$, $b = 0_b = (0, 0, 0)$, $c = 0_c = (0, 0, 0)$, $d = 0_d = (0, 0, 0)$ である場合には、(数式24)の2電子積分は、図28の(数式26)～(数式34)に示すように求まる。

【0039】ここで、(数式26)に現れる $[\cdots, \cdots]^{(m)}$ は補助積分、 m は補助インデックスであるが、これらについては後で述べる。(数式27)の積分範囲は、0から1である。

【0040】また、 a, b, c, d のうち1つでも s 軌道以外のものがある場合には、図29の(数式35)および(数式36)に示す漸化式を用いて計算する。

【0041】(数式35)で、添え字の i は、 x または y または z 成分であることを示す。また、 1_i は、 i 成分のみ1で、他は0であるようなベクトルである。さらに、 $N_i(n)$ は、角運動量 n の i 成分の値を示すものである。(数式35)は、左辺の補助積分に現れる角運動量の1つは右辺では確実に1以上減少し、また、左辺の補助積分の補助インデックスは右辺では同じあるいは1だけ増加する、という性質を有している。

【0042】補助積分 $[\cdots, \cdots]^{(m)}$ は、補助インデックス m が0であるときに、2電子積分 $[\cdots, \cdots]$ に正確に一致するものであり、2電子積分の計算を補助するものである。どんなに角運動量の大きな基底関数を含んだ2電子積分であっても、(数式35)を繰り返

10

20

30

40

50

返し用いて角運動量を減少させ、最後には、全て角運動量が $(0, 0, 0)$ であるような補助積分に行き着くことができる。角運動量が $(0, 0, 0)$ であるような補助積分は、(数式 26) を用いて計算できるので、その数値と適当な係数を乗じ、加算すれば 2 電子積分の数値が得られる。

【0043】実際の計算は、以下のように行う。まず、(数式 35) に従って、2 電子積分を 8 つ以下の補助積分を用いた形式に表わす。ここで現れた補助積分に対して、さらに、(数式 35) を適用する。このような手続きを繰り返して、全て角運動量が $(0, 0, 0)$ であるような補助積分に行き着くまでの道筋を、計算手順として記録しておく。

【0044】次に、(数式 26) を用いて角運動量が $(0, 0, 0)$ であるような補助積分を計算し、そこから出発して、先ほどの計算手順をたどりながら補助積分の数値を計算していき、最後に目的とする 2 電子積分の数値を得る。

【0045】(数式 35) が有するもう一つの重要な性質は、2 電子積分に現れる 4 つの角運動量の組み合わせが同じであれば、軌道指数や原子核座標の組み合わせが異なっている、上記の計算手順としては全く同じものを用いることができることである。計算の実行に際しては、軌道指数や原子核座標に応じて補助積分に乗じる係数を変えてやるだけで良い。

【0046】[カットオフ] 上述したように、計算しなければならない縮約基底関数で表わした 2 電子積分の数は、縮約基底関数の数 N に対して N^4 となる。実際に数値を得なければならないのは、原始基底関数で表わした 2 電子積分の方であるが、その総数は、縮約基底関数で表わした 2 電子積分の数の数倍から数 10 倍 (縮約基底関数を構成する原始基底関数の数、すなわち縮約数に依存する) に及ぶ。

【0047】この個数を減らす手法として、第 1 に考えられるのは、(数式 19) あるいは (数式 25) に記した対称性を利用することである。しかしながら、この方法では最も効率化を行っても 2 電子積分の計算量は $1/8$ にしかならない。

【0048】もう 1 つの手法は、計算精度の観点から、不必要と判断できる 2 電子積分の計算を、積極的に排除する方法である。不必要な 2 電子積分の判断は、以下のように行うことができる。

【0049】上述したように、全ての 2 電子積分の数値は、(数式 26) に示した全て角運動量が $(0, 0, 0)$ であるような補助積分 $[00, 00]$ ^(m) の数値に基づいて計算される。したがって、2 電子積分の数値の、フォック行列要素の数値への寄与が計算誤差の程度であるかどうかを、 $[00, 00]$ ^(m) の数値で判断することが可能である。さらに、 $[00, 00]$ ^(m) の数値の大きさは、(数式 29) に示した関数 $K(\xi,$

$\xi', R, R')$ の値から、さらに、それは、図 29 の (数式 37) の大きさから判断することができる。

【0050】したがって、 (ξ_a, A, ξ_b, B) の数値の組み合わせで、(数式 25) の 1 つ目の関数 K の大きさを見積ることで、2 電子積分 $[ab, **]$ を計算する必要があるかどうかを判断し、また、 (ξ_c, C, ξ_d, D) の数値の組み合わせで、(数式 26) の 2 つ目の関数 K の大きさを見積ることで、2 電子積分 $[*, *, cd]$ を計算する必要があるかどうかを判断することができる。

【0051】このようにして、不必要な 2 電子積分の計算を排除することを、「カットオフする」と呼ぶことにする。上記の例で、 a および b の情報だけから判断してカットオフする場合には、 a, b でのカットオフ、 c および d の情報だけから判断してカットオフする場合には、 c, d でのカットオフ、と呼ぶ場合がある。このように、 a, b だけで、あるいは c, d だけでカットオフするかどうかの判断ができるのは、図 29 の (数式 37) の最大値が 1 で下限値が 0 だからである。このようにカットオフを行うことにより、計算しなければならない 2 電子積分は、概略で N^2 に比例する個数となり、計算量を大幅に低減できる。

【0052】上述のことから、 N が大きい場合には、2 電子積分の対称性を利用することによる効果よりも、カットオフによる計算量低減の効果の方が桁違いに大きく、これを取り入れることによって、非経験的分子軌道計算におけるフォック行列の計算に要する処理時間が大きく短縮できることがわかる。

【0053】[分子軌道計算機システムの例] フォック行列要素の計算を、並列計算機を用いて高速に行うシステムの例として、文献 2 (白川他, "超高速分子軌道計算専用機 MOE のアーキテクチャ", 電子情報通信学会技術報告, vol. CPSY96-46, no. 5, pp. 45-50, 1996) に記載のシステムがある。

【0054】この文献 2 では、ホスト計算機に、複数個のプロセッサエレメントをバスを介して接続して並列処理を行うシステムが示されている。この文献 2 では、このような構成を有する並列処理システムのアーキテクチャの検討に際して、 R, S, T, U の 4 つのインデックスで構成される 4 重ループのまわし方および並列化を行う部分の種々の方法に関して、全体の計算量およびプロセッサエレメントに必要なメモリ量を見積っている。

【0055】文献 2 に記載されている並列処理システムは、個々のプロセッサ・エレメントが高い演算処理能力を有する上、システム全体を低価格で実現することが可能であるため、コストパフォーマンスの優れた計算システムを提供することができる。しかしながら、文献 2 では、前述したカットオフを考慮する場合の方法や具体的なループの制御方法への言及がなく、効率的な処理が行えるかどうか不明であった。

【0056】[I. Fosterらの方法] フォック行列要素の計算を、並列計算機を用いて効率的に行うアルゴリズムとして、文献3 (I. T. Foster, et al., "Toward High-Performance Computational Chemistry: I. Scalable Fock Matrix Construction Algorithms", Journal of Computational Chemistry, vol. 17, no. 1, p. 109, 1996) に記載のアルゴリズムがある。

【0057】この文献3では、幾つかのフォック行列要素計算アルゴリズムに関して、その計算量およびホスト計算機と、複数のプロセッサ・エレメントとの間の通信量を解析している。その内容を、以下に説明する。

【0058】第1のアルゴリズムは、カノニカル法と呼ばれる最も簡単なアルゴリズムである。この手法では、1つのプロセッサ・エレメントに、図30に示す(数式38)の関係を満たす4個の縮約基底関数I, J, K, Lと、6個の密度行列要素P_{IJ}, P_{IK}, P_{IL}, P_{JK}, P_{JL}, P_{KL}とを渡し、プロセッサエレメントには2電子積分を計算させて、フォック行列要素F_{IJ}, F_{IK}, F_{IL}, F_{JK}, F_{JL}, F_{KL}の一部を、図30の(数式39)に従って計算させる。

【0059】1つの2電子積分を計算する間に、ホスト計算機とプロセッサエレメントとの間で通信される行列要素の数を、perERIという単位で勘定することになると、この場合には、通信データ数が12 [perERI] となる。

【0060】第2のアルゴリズムは、トリプルソート法と呼ばれるアルゴリズムである。図31の(数式40)の関係を満たす4個の縮約基底関数I, J, K, Lと、6個の密度行列要素P_{IJ}, P_{IK}, P_{IL}, P_{JK}, P_{JL}, P_{KL}とを、1つのプロセッサエレメントに渡し、そのプロセッサエレメントに3つの2電子積分G(I, J, K, L), G(I, K, J, L), G(I, L, J, K)を計算させて、フォック行列要素F_{IJ}, F_{IK}, F_{IL}, F_{JK}, F_{JL}, F_{KL}の一部を、図31の(数式41)に従って計算させる。

【0061】この場合、3つの2電子積分を計算する間に、6つの密度行列要素と、6つのフォック行列要素の転送が必要であるので、通信データ数は、4 [perERI] である。したがって、通信データ数の観点からカノニカル法より優れていると言える。

【0062】しかしながら、プロセッサエレメントにおける原始基底関数で表わした2電子積分の計算時間を、例えば2マイクロ秒(=10⁻⁶秒、以下ではμsと記す)と仮定し、また、縮約基底関数の平均の縮約数を1.5と仮定し、密度行列要素やフォック行列要素が、倍精度の浮動小数点数すなわち64ビットのデータサイ

ズであると仮定すると、1つの縮約基底関数で表わした2電子積分の計算時間は、約10μsとなり、1つのプロセッサエレメント当たりで、25.6Mbps (25.6×10⁶ bit per second) の通信性能が、ホスト計算機とプロセッサエレメントとの間で必要とされることになる。

【0063】計算性能を上げるために、プロセッサエレメント数Mを、例えば100とした場合には、全体の通信性能としては、2560Mbpsが要求されることとなるが、現在の技術で、このような通信性能を達成するのは容易でない。

【0064】安価な通信手段、例えばIEEE1394バス規格で定められたシリアル通信では、200Mbpsの通信性能が実現できるが、それを用いてトリプルソート法を採用したフォック行列要素並列計算を行うと、全体の処理時間が通信時間で律速されてしまい、行列の対称性を利用した計算時間低減の効果は得られなくなってしまう。

【0065】第3のアルゴリズムは、単純ブロック法と呼ばれる手法である。これは、さらに、カノニカル法に基づいたものと、トリプルソート法に基づいたものとに細分類できる。この第3のアルゴリズムは、縮約基底関数をブロック化しておくことにより、密度行列要素やフォック行列要素の利用効率を高め、通信量を低減する手法である。

【0066】トリプルソート法に基づいて、その手法を説明する。まず、N個ある縮約基底関数を、I_C 個毎に、n (=N÷I_C) 個のブロックに分割する。ブロックの番号を、I_B, J_B, K_B, L_B のように表わすことにする。次に、図31の(数式42)の関係を満たす4個の縮約基底関数ブロックI_B, J_B, K_B, L_B と、6個の密度行列要素ブロックP(I_B, J_B), P(I_B, K_B), P(I_B, L_B), P(J_B, K_B), P(J_B, L_B), P(K_B, L_B) を、1つのプロセッサエレメントに渡す。渡される密度行列要素の数は6 I_C² 個となる。

【0067】プロセッサエレメントが計算する2電子積分は、G(I, J, K, L), G(I, K, J, L), G(I, L, J, K)に対応する3 I_C⁴ 個の2電子積分であり、プロセッサエレメントは、前述の(数式41)と同様に、フォック行列要素ブロックF(I_B, J_B), F(I_B, K_B), F(I_B, L_B), F(J_B, K_B), F(J_B, L_B), F(K_B, L_B) を計算して、ホスト計算機に送り返す。

【0068】このときに送り返されるフォック行列要素数も、6 I_C² 個である。その結果、通信データ数は、12 I_C² ÷ 3 I_C⁴ = 4 / I_C² [perERI] となる。つまり、ブロック内の縮約基底関数の数を多くすればするほど、密度行列要素やフォック行列要素の利用効率が高まり、通信量が低減する。なお、カノニカル法

の場合には、図 31 の (数式 42) に代わり (数式 43) を用いる。

【0069】さらに、通信量を低減する第 4 の手法として、列ブロック法がある。これは、単純ブロック法において、 I_B 、 J_B 、 K_B の組み合わせが同じで、 L_B だけが異なる計算を、全て 1 つのプロセッサエレメントに割り当てる手法である。この手法のフローチャートを図 20 に示した。なお、図 20 において破線で示した矢印の部分は、その後の処理がその前の処理で律速されるのではなく、他の処理系からの情報の入力待ちとなることを示している。

【0070】列ブロック法も、さらに、カノニカル法に基づいたものとトリプルソート法に基づいたものとに細分類できるが、トリプルソート法に基づいて、図 20 のフローチャートを参照しながら説明する。

【0071】まず、最初に単純ブロック法と同様に、 N 個ある縮約基底関数を、 I_C 個毎に $n (=N \div I_C)$ 個のブロックに分割する (ステップ S1)。次に、ホスト計算機は、特定のプロセッサエレメントに割り当てる縮約基底関数ブロック I_B 、 J_B 、 K_B の組み合わせ (I_B 、 J_B 、 K_B) を決定する (ステップ S2)。

【0072】次に、ホスト計算機は、プロセッサエレメントへ、前述の (数式 42) の関係を満たす 3 個の縮約基底関数ブロック I_B 、 J_B 、 K_B に対応して、要素数が各々 $I_C \times I_C$ 個の密度行列要素ブロック $P(I_B, J_B)$ 、 $P(I_B, K_B)$ 、 $P(J_B, K_B)$ および要素数が各々 $K_B \times I_C \times I_C$ 個の密度行列要素ブロックの列 $P(I_B, L)$ 、 $P(J_B, L)$ 、 $P(K_B, L)$ を送信する (ステップ S3 およびステップ S4)。但し、 L は、1 から $K_B \times I_C$ の範囲の全てである。

【0073】次に、これらをステップ S11 およびステップ S12 で受信したプロセッサエレメントは、内部で L に関するループをまわし、単純ブロック法と同じ 2 電子積分およびフォック行列要素を計算する (ステップ S13)。全ての L に対する計算が終了すると、プロセッサエレメントは、要素数が各々 $K_B \times I_C \times I_C$ 個のフォック行列要素ブロックの列 $F(I_B, L)$ 、 $F(J_B, L)$ 、 $F(K_B, L)$ および要素数が各々 $I_C \times I_C$ 個のフォック行列要素ブロック $F(I_B, J_B)$ 、 $F(I_B, K_B)$ 、 $F(J_B, K_B)$ をホスト計算機に送り返す (ステップ S14 およびステップ S15)。

【0074】ホスト計算機は、ステップ S5 およびステップ S6 で、それらを受信する。そして、ステップ S2 に戻り、以上の処理を繰り返す。

【0075】このようにすることにより、密度行列要素やフォック行列要素の利用効率がさらに高まり、2 電子積分 1 つあたりの通信データ数は、 $2/N I_C + 2/I_C^2$ [perERI] となり、 $N \gg I_C$ を仮定すれば、単純ブロック法の約半分となる。

【0076】さらに、プロセッサエレメント上の I_B 、 J_B 、 K_B の組み合わせを更新した際に、変化するのが K_B のみである場合には、行列要素ブロック $P(I_B, J_B)$ 、 $F(I_B, J_B)$ および行列要素ブロックの列 $P(I_B, L)$ 、 $P(J_B, L)$ 、 $F(I_B, L)$ 、 $F(J_B, L)$ は、プロセッサエレメント上に残して再利用できるので、通信データ数はさらに減って、 $4/3 N I_C + 2/3 I_C^2$ [perERI] となる。

【0077】列ブロック法と同様の思想で、さらに密度行列要素やフォック行列要素の利用効率を高める第 5 の手法として、クラスタリング法が、文献 2 では紹介されている。しかしながら、この手法は、負荷分散あるいはスケラビリティの観点から劣る手法であるとされており、ここでは説明を省略する。

【0078】

【発明が解決しようとする課題】 [カットオフを考慮した場合の問題点] 上述した文献 3 に記載されているうちで、最も優れた第 4 のアルゴリズムであっても、カットオフを考慮した場合には不都合が生じる場合がある。そのような例を以下に示す。なお、カットオフにより生き残る割合を α 、プロセッサエレメント数を M 、1 つの 2 電子積分 $G(I, J, K, L)$ 当たりの計算時間を $T_{eri} (\mu s)$ 、行列要素のデータ長を 64 ビットとする。

【0079】カノニカル法の場合には、1 つのプロセッサエレメントに割り当てられたジョブ当たりに発生する通信量は、最低 (すなわち、組み合わせの更新時に変化するのが K_B だけの場合) でも、

$$2(2I_C^2 + K_B I_C^2) \times 64 \quad (\text{bit})$$

となる一方、その間にプロセッサエレメントで計算される 2 電子積分の個数は、

$$\alpha^2 K_B I_C^4 \quad (\text{個})$$

なので、全体に必要な通信性能は、図 32 に示す (数式 44) のようなものとなる。

【0080】トリプルソート法の場合には、1 つのプロセッサエレメントに割り当てられたジョブ当たりに発生する通信量は、最低 (すなわち、組み合わせの更新時に変化するのが K_B だけの場合) でも、

$$2(2I_C^2 + K_B I_C^2) \times 64 \quad (\text{bit})$$

となる一方、その間にプロセッサエレメントで計算される 2 電子積分の個数は、

$$3\alpha^2 K_B I_C^4 \quad (\text{個})$$

なので、全体に必要な通信性能は、図 32 に示す (数式 45) のようなものとなる。

【0081】並列処理のプロセッサ・エレメント数 M を 100、プロセッサエレメントにおける原始基底関数で表わした 2 電子積分 $g(i, j, k, l)$ の計算時間が $2 \mu s$ 、縮約基底関数の平均縮約数を 1.5 (従って、2 電子積分 $G(I, J, K, L)$ 1 つ当たりの計算時間 T_{eri} を $10 \mu s$)、カットオフにより生き残る割合

α を 0.05 とした場合に、必要とされるホスト計算機とプロセッサエレメントとの間の通信性能のブロックサイズ I_C に対する依存性を計算すると、カノニカルの場合には、図 21 に示すように、トリプルソートの場合には、図 22 に示すようになる。

【0082】カノニカルの場合にも、また、トリプルソートの場合にも、必要な通信性能は K_B の値に依存して変化するが、 $K_B > 100$ ではその変化量が小さく、実際の通信性能に応じて、ブロックサイズ I_C を、カノニカルの場合なら、例えば 50 に、トリプルソートの場合であれば、例えば 30 に、それぞれ設定することが可能である。

【0083】文献 3 で前提としているような、十分に大きなデータ保持容量を有するワークステーションを、多数台用いるシステムでは、このような方法で計算を行うことが可能であるが、そのようなシステムを構成するには、多大な費用が必要である。

【0084】一方、文献 2 にあるような、せいぜい数 10 M ビット程度の小さな容量のメモリが接続される安価な専用プロセッサ・エレメントを用いた計算システムでは、列単位で保持できる行列要素は 10 列以下、すなわち、許されるブロックサイズは 2 から 3 程度までである。

【0085】この場合には、図 21 あるいは図 22 の結果から、通信の手段として IEEE 1394 バス規格で定められているような安価なシリアル通信を用いた場合には、その性能が 200 Mbps であることを考慮すると、通信性能が律速しない効率的な処理を行うことが不可能である。

【0086】この発明は、以上のような点にかんがみ、安価な通信手段と小容量のメモリを有する多数のプロセッサ・エレメントを用いた並列計算によっても、ホスト計算機とプロセッサ・エレメントとの間の通信性能に律速されることなく、効率的に行列要素計算を行える並列処理方法を提供することを目的とする。

【0087】

【課題を解決するための手段】上記目的を達成するために、請求項 1 の発明の並列処理方法は、同じ 1 から N (N は正の整数) の範囲にある 4 つの整数型の変数 R, S, T, U を用いて表わされ、 $G(R, S, T, U) = G(R, S, U, T) = G(S, R, T, U) = G(S, R, U, T) = G(T, U, R, S) = G(T, U, S, R) = G(U, T, R, S) = G(U, T, S, R)$ なる関係を満たす関数 G の関数値 $G(R, S, T, U)$ と；2 つの変数 T, U を用いて表わされ、 $P(T, U) = P(U, T)$ なる関係を満たす行列 P の要素 $P(T, U)$ と；係数 $A1$ との積 $A1 \cdot P(T, U) \cdot G(R, S, T, U)$ についての前記範囲の全ての T および U に関する総和 $F1(R, S)$ と、前記関数値 $G(R, U, T, S)$ と；前記行列要素 $P(T, U)$ と；

係数 $A2$ との積 $A2 \cdot P(T, U) \cdot G(R, U, T, S)$ に関する前記範囲の全ての T および U における総和 $F2(R, S)$ との和 $F(R, S) = F1(R, S) + F2(R, S)$ を要素とする行列 F の全要素の計算を、ホスト計算機と、1 つまたは複数個のプロセッサエレメントとを有する並列処理装置を用いて行う並列処理方法において、前記変数 R, S, T, U について、3 重ループを形成し、前記 3 重ループの最も外側のループは、 $R \leq N$ および $T \leq R$ なる関係を満たす変数 R と変数 T との組み合わせに関するループとし、前記最も外側のループの内側の 2 番目は前記変数 S に関するループ、前記 2 番目よりも内側の 3 番目は前記変数 U に関するループとするか、あるいは前記 2 番目は変数 U に関するループ、前記 3 番目は変数 S に関するループとし、前記変数 S のとりうる値の範囲を 1 から R の間とし、前記変数 U のとりうる値の範囲を 1 から R の間とし、前記 3 番目のループの内側で、所定の前記関数値 $G(R, S, T, U)$ の計算およびその計算結果を用いた所定の前記行列要素 F の一部の計算を行うものであって、前記 2 番目および 3 番目のループをひとまとまりとして 1 つのジョブ単位を形成し、前記ジョブ単位毎に前記複数のプロセッサエレメントに処理させることを特徴とする。

【0088】また、請求項 2 の発明の並列処理方法は、 N 個 (N は正の整数) の縮約シェルを用いて表現される分子のエネルギーを計算する分子軌道計算を、ホスト計算機と、1 つまたは複数個のプロセッサエレメントとを有する並列処理装置を用いて行う並列処理方法において、縮約シェル R, S, T, U のそれぞれに含まれる原始シェル r, s, t, u のそれぞれの成分である原始基底関数 i, j, k, l をインデックスとして用いて表わされる 2 電子積分関数 g の関数値 $g(i, j, k, l)$ と；前記原始基底関数 k をひとつの構成要素とする縮約基底関数 K および前記原始基底関数 l をひとつの構成要素とする縮約基底関数 L とをインデックスとして用いて表わされる密度行列 P の要素 $P(K, L)$ と；係数 $A1$ との積 $A1 \cdot P(K, L) \cdot g(i, j, k, l)$ の全ての縮約基底関数に関する総和 $f1(I, J)$ と、前記 2 電子積分関数の関数値 $g(i, k, j, l)$ と；前記密度行列 P の要素 $P(K, L)$ と；係数 $A2$ との積 $A2 \cdot P(K, L) \cdot g(i, k, j, l)$ の全ての縮約基底関数に関する総和 $f2(I, J)$ との和 $f(I, J) = f1(I, J) + f2(I, J)$ の、前記原始基底関数 i, j をひとつの構成要素とする前記縮約基底関数 I, J に含まれる全ての前記原始基底関数に関する和で表わされるフォック行列 F の全ての行列要素 $F(I, J)$ の計算を、最も外側のループは、 $R \leq N$ および $T \leq R$ なる関係を満たす前記縮約シェル R と T との組み合わせに関するループとし、前記最も外側のループの内側の 2 番目は前記縮約シェル S に関するループ、前記 2 番目よりも内側の 3 番目は前記縮約シェル U に関するループ

とするか、あるいは前記 2 番目は前記縮約シェル U に関するループ、前記 3 番目は前記縮約シェル S に関するループとし、前記縮約シェル S のとりうる値の範囲を 1 から R の間とし、前記縮約シェル U のとりうる値の範囲を 1 から R の間とし、前記 3 番目のループの内側で、所定の 2 電子積分の計算およびその結果を用いた所定のフォック行列要素の一部の計算を行うものであって、前記 2 番目および 3 番目のループをひとまとまりとして 1 つのジョブ単位を形成し、前記ジョブ単位毎に前記複数のプロセッサ・エレメントに処理を割り当てることを特徴とする。

【0089】

【作用】上述の構成の請求項 1 の発明、請求項 2 の発明を用いて、フォック行列計算アルゴリズムを行うと、カットオフを考慮して 2 電子積分の計算数を減少させる場合においても、ホスト計算機とプロセッサエレメントとの間の通信性能およびホスト計算機の処理性能に律速されることのない効率的な並列計算によって、フォック行列を高速に計算することができる。

【0090】

【発明の実施の形態】以下、この発明の実施の形態を、図を参照しながら説明する。以下に説明する実施の形態では、図 1 に示すような安価な計算機システムを用いる。

【0091】すなわち、図 1 は、この実施の形態の並列計算システムの全体の構成を示すもので、ホスト計算機 1 に対して、バス 3 を通じて、複数のプロセッサエレメント 2 が接続されている。バス 3 としては、例えば IEEE 1394 シリアルバスが用いられる。

【0092】なお、各プロセッサエレメント 2 のメモリ容量は、数 10 M ビット、例えば 20 M ビットであり、行列要素が 64 ビットの浮動小数点数で表わされる密度行列およびフォック行列のサイズが、 10000×10000 であっても、各行列の 10 列分は十分格納できる容量のものが用いられる。この程度の容量のメモリを各プロセッサエレメントに持たせることは、現在の技術で十分に可能である。

【0093】まず、この実施の形態における行列要素計算方法と比較するために、従来例のトリプルソート法における全体の計算アルゴリズムを、プログラムコードの形式で表わしたものを図 2 に示す。

【0094】図 2 で、 N_{shell} は縮約シェルの総数を表わす整数型の変数、 R 、 S 、 T 、 U は縮約シェル番号に用いる整数型の変数、 I 、 J 、 K 、 L は縮約基底番号に用いる整数型の変数、 G_{IJKL} 、 G_{IKJL} 、 G_{ILJK} は 2 電子積分の数値に用いる実数型の変数、 F 、 P はフォック行列および密度行列に用いる実数型の 2 次元配列である。

【0095】また、ここでは、縮約基底が通し番号で番号付けされていること、および同一の縮約シェルを構成

する縮約基底の番号が連続していることを前提として、 b_basis および e_basis は、その引数となっている番号に相当する縮約シェルを構成する縮約基底の番号の始まりと終わりをリターン値とする整数型の関数である。さらに、 G は、その引数となっている 4 つの番号 I 、 J 、 K 、 L に対応する縮約基底で一意的に決まる 2 電子積分を、前述の (数式 23) に従って計算する実数型の関数である。

【0096】さて、この図 2 に示す従来例のトリプルソート法では、縮約シェルの計算に関する 4 重ループのうちの最も内側で行われるフォック行列要素への足し込みの計算において、密度行列要素 $P[I][J]$ 、 $P[I][K]$ 、 $P[I][L]$ 、 $P[J][K]$ 、 $P[J][L]$ 、 $P[K][L]$ が用いられている。

【0097】したがって、縮約シェルに関する 4 重ループをどのような順序で形成しても、ホスト計算機 1 から密度行列要素がプロセッサ・エレメント 2 に送信される頻度を、1 つの 2 電子積分計算あたりの行列要素数で表わすと、それは 1 のオーダー (トリプルソート法では正確には $1/3$) となる。

【0098】前述した従来例におけるブロック法は、行列要素の再利用性を高めて密度行列要素が送信される頻度を 1 より小さい定数倍にするものであった。ここで、その定数は、ブロックサイズの 2 乗に反比例するものである。図 1 に示すような安価なシステムを前提とした場合には、プロセッサエレメント 2 のメモリの容量の大きさで上限が規定されて、ブロックサイズを大きくできないため、密度行列要素が送信される頻度が、1 よりも大して小さくならなかった。

【0099】また、計算の結果として得られるフォック行列要素は、 $F[I][J]$ 、 $F[I][K]$ 、 $F[I][L]$ 、 $F[J][K]$ 、 $F[J][L]$ 、 $F[K][L]$ であり、プロセッサ・エレメント 2 からホスト計算機 1 へフォック行列要素が送信される頻度は、密度行列の場合と全く同様であった。さらに、カットオフを考慮した場合には、カットオフにより生き残る割合 α の 2 乗に反比例して通信の頻度が高くなるという問題があった。

【0100】この発明の実施の形態のアルゴリズムでは、カットオフにより生き残る割合 α に対する通信の頻度の依存性を弱くし、通信量を低く保つことが可能である。この実施の形態の計算アルゴリズムをプログラムコードの形式で表わしたものを図 3 に示す。

【0101】すなわち、図 3 のアルゴリズムでは、最も外側のループは $R \leq N_{shell}$ および $R \geq T$ なる関係を満たす縮約シェル R と T との組み合わせ (RT) に関するループとしている。そして、2 番目は縮約シェル S に関するループ、3 番目は縮約シェル U に関するループとしている。この場合、 S のとりうる値の範囲は 1 から R の間とし、また、 U のとりうる値の範囲も 1 から R の

間とする。さらに、3番目のループの内側で、関数値 $G(R, S, T, U)$ の計算およびその結果を用いた所定の行列要素 F の一部の計算を行うようにしている。

【0102】なお、2番目は U に関するループ、3番目は S に関するループとするようにしてもよい。すなわち、図3では、 U に関するループが、 S に関するループの内側に形成されているが、このループの順序は逆でも良い。

【0103】そして、 R と T とが固定された2番目および3番目の2つのループをひとまとまりとして一つのジョブ単位を形成し、このジョブ単位毎に複数のプロセッサエレメント2に処理させるようにする。

【0104】このとき、ホスト計算機1は、全ての行列要素 P の初期値の計算と、プロセッサエレメントと共に
10 行う SCF 計算と、この SCF 計算を継続するかどうかの決定などを行う。 SCF 計算に当たって、ホスト計算機1は、複数個のプロセッサエレメント2に対する R と T とが固定されたジョブ単位の割り当て決定と、プロセッサエレメントに対して送信すべき行列 P の一部の行列要素の選択と、選択された行列要素のプロセッサエレメントに対する送信と、プロセッサエレメントから送信されてきた行列 F の一部の行列要素の受信と、行列 F を用いた行列 P の更新とを行う。

【0105】一方、プロセッサエレメント2は、ホスト計算機から送信された行列 P の一部の行列要素の受信と、縮約シェル S に関するループの制御および縮約シェル U に関するループの制御と、関数 $G(R, S, T, U)$ または関数 $g(i, j, k, l)$ の計算と、行列 F の一部の行列要素の計算と、行列 F の一部の行列要素の
20 ホスト計算機に対する送信とを行う。

【0106】この実施の形態では、ホスト計算機1とプロセッサ・エレメント2との間で通信される密度行列要素およびフォック行列要素を、 $P[I][J]$, $P[I][L]$, $P[J][K]$, $P[K][L]$ および $F[I][J]$, $F[I][L]$, $F[J][K]$, $F[K][L]$ だけにし、また、プロセッサ・エレメント2で計算する2電子積分を、 G_{IJKL} だけにした。この実施の形態の場合、行列要素の通信頻度は、 $1/N$ のオーダーである。

【0107】ここで、図3の計算アルゴリズムで行列要素の通信頻度を $1/N$ のオーダーとすることが可能な理由を説明する。

【0108】前述したように、図3において、最も外側のループは、縮約シェル R と T の組み合わせ（以下では RT ペアと記す）の番号 RT に関するループである。 R と T の組み合わせの総数は、 $N_{shell} \times (N_{shell} + 1) / 2$ である。以下の説明では $R \geq T$ を前提とする。したがって、 R の取りうる範囲は、1から N_{shell} まで、 T の取りうる範囲は、1から R までである。

【0109】図3では、表記を簡略するために、 RT に関しては、1から $N_{shell} \times (N_{shell} + 1) / 2$ まで、 S および U に関しては、1から R まで、それぞれ番号の小さい順番にループを回すようにしているが、必ずしもこのような順序でループを回す必要はない。また、 S および U に関しては、1から R までの全ての値を取る必要はない。

【0110】縮約シェル U に関するループの内部では、縮約基底 I, J, K, L に関する4重ループとなっており、その内部で2電子積分の計算を行い、また、 I, J, K, L 相互の関係に依存した条件に従ったフォック行列の足し込み計算を行なうアルゴリズムとなっている。上記の条件分岐により、(数式15)に示したフォック行列要素の計算を、過不足なく計算することが可能となる。

【0111】前述したように、図3のアルゴリズムで、縮約シェル R および T に関するループの制御はホスト計算機1で行い、縮約シェル S に関するループより内側はプロセッサエレメント2で行う。すなわち、各プロセッサエレメント2には、縮約シェル R と T の組み合わせ
20 (RT ペア) が固定されたジョブが割り当てられる。ある RT ペアが固定されたジョブが割り当てられたプロセッサ・エレメント2において、使用する密度行列要素は、 $P[I][J]$, $P[I][L]$, $P[J][K]$, $P[K][L]$ である。

【0112】ここで、 I および K は、それぞれ縮約シェル R および T を形成する縮約基底であるので、ジョブが割り当てられた時点で、これらは数個（縮約シェルが s 軌道の場合には1個、 p 軌道の場合には3個、 d 軌道の場合には6個）に固定される。 J および L は任意である。

【0113】したがって、 RT ペアに関するジョブのために、ホスト計算機1からプロセッサエレメント2に送信する密度行列要素数は、 N の定数倍のオーダーである。フォック行列に関しても同様である。プロセッサエレメント2には、これら全てを保持することができるだけの容量のメモリを有している。

【0114】一方、 RT ペアが固定されたジョブが割り当てられたプロセッサエレメントでは、縮約シェル S および U に関するループが回るので、そこで計算される2電子積分の個数は、 N^2 のオーダーとなる。したがって、1つの2電子積分当たりに換算した通信の頻度は $1/N$ となる。

【0115】このように、複数のプロセッサエレメント2へのジョブ割り当てが、 RT ペア単位で行われることから、この発明のアルゴリズムを、 RT 並列アルゴリズムと呼ぶことにする。

【0116】 RT 並列アルゴリズムでカットオフを考慮する場合には、プロセッサエレメント2で計算する2電子積分の個数は、カットオフにより生き残る割合 α の2
50

乗に比例して減少する。一方、計算に用いる密度行列要素および計算されるフォック行列要素も、カットオフにより生き残る割合 α に比例して減少させることができる。この理由を以下に説明する。

【0117】プロセッサエレメント2に割り当てるRTペアを決めた時点で、ホスト計算機1は、IJペアでカットオフされないJ、およびKLペアでカットオフされないLをリストアップできる。

【0118】したがって、ホスト計算機1は、Iが固定された1列の密度行列要素P[I][J]の中からIJペアでカットオフされないJに対応するものだけを、Kが固定された1列の密度行列要素P[K][J]の中からIJペアでカットオフされないJに対応するものだけを、Iが固定された1列の密度行列要素P[I][L]の中からKLペアでカットオフされないLに対応するものだけを、Kが固定された1列の密度行列要素P[K][L]の中からKLペアでカットオフされないLに対応するものだけを、それぞれ選り出して送信することが可能である。そのため、ホスト計算機1から送信される密度行列要素数は、 α に比例して減少する。

【0119】また、プロセッサエレメント2は、Iが固定された1列のフォック行列要素F[I][J]のうち、IJペアでカットオフされないJに対応するものだけを、Kが固定された1列のフォック行列要素F[K][J]のうち、IJペアでカットオフされないJに対応するものだけを、Iが固定された1列のフォック行列要素F[I][L]のうち、KLペアでカットオフされないLに対応するものだけを、Kが固定された1列のフォック行列要素F[K][L]のうちKLペアでカットオフされないLに対応するものだけを、それぞれ計算する。

【0120】プロセッサエレメント2からホスト計算機1へは、計算されたフォック行列要素のみを送信すればよいので、プロセッサエレメント2からホスト計算機1へ送信されるフォック行列要素数も、カットオフにより生き残る割合 α に比例して減少する。

【0121】したがって、カットオフを考慮した計算を行う場合においては、1つの2電子積分あたりに換算した通信の頻度は、カットオフにより生き残る割合 α に反比例して増加するが、それは、 α の2乗に反比例して増加する従来例と比較して、実用上の大きな利点となる。

【0122】次に、プロセッサエレメントの数をM、1つの2電子積分G(I, J, K, L)あたりの計算時間をTeri(μ s)、1つの縮約シェルを構成する平均の縮約基底数をa、行列要素のデータ長を64bitとして、必要な通信性能を定式化する。

【0123】密度行列要素ブロックP(R, S)が、縮約シェルRを構成する任意の縮約基底Iと縮約シェルSを構成する任意の縮約基底Jとの組み合わせで表わされる密度行列要素P(I, J)の集合であるとする、そ

の密度行列要素ブロックの要素数は、約 a^2 となる。密度行列要素ブロックP(R, S)は、S=1から、S=Rまでの範囲のうち、RSの組み合わせでのカットオフで生き残るもの全てが、ホスト計算機1からプロセッサエレメント2へ転送される。したがって、その要素数は、 $a^2 \times aR \times \alpha$ 個となる。

【0124】他の密度行列要素ブロックP(R, U), P(T, S), P(T, U)に関しても、概略で同じ要素数となる。また、プロセッサエレメント2からホスト計算機1へ送信するフォック行列要素数は、密度行列要素数と全く同じである。したがって、1つのジョブ当たりのホスト計算機1とプロセッサエレメント2との間の通信データ量は、
 $8 \alpha a^3 R \times 64$ (ビット)
 となる。

【0125】一方、1つのジョブ当たりで計算される2電子積分の個数は、概略で、
 $\alpha^2 a^4 R^2$ (個)
 なので、全体に必要な通信性能は、図32の(数式46)に示すようなものと見積もられる。

【0126】プロセッサエレメント2の数Mを100、プロセッサエレメント2における原始基底関数で表わした2電子積分G(I, J, K, L)あたりの計算時間Teriを10(μ s)、1つの縮約シェルを構成する平均の縮約基底数aを3、カットオフによる生き残り割合 α を0.05とした場合に、必要な通信性能のRに対する依存性を図4に示す。

【0127】この図4からわかるように、縮約シェルRが小さいごく一部の範囲では、必要な通信性能が100Mbpsを超えるが、他の殆どの領域では100Mbpsの通信性能があれば、十分に通信時間を計算時間でカバーできることがわかる。

【0128】一定のRに対するジョブ数が、 R^2 に比例することを考慮すると、通信時間が計算時間を上回る可能性は、ごくわずかとなる。したがって、この実施の形態のRT並列アルゴリズムは、100Mbps程度の通信性能の、安価な通信手段を用いても、システム全体の処理効率を低下させることのない、効率的なアルゴリズムとなっている。

【0129】[より具体的な説明] 以下では、この発明の実施の形態の並列処理方法の詳細を、より具体的に説明する。

【0130】[ホスト計算機1の処理手順] 図5は、この発明の実施の形態の並列処理方法を説明するフローチャートである。ホスト計算機1とプロセッサエレメント2とは異なる処理を行うため、それぞれについてフローチャートを併記した。

【0131】また、プロセッサエレメント2は、並列に複数(典型的には100個)がホスト計算機1に接続されていることを前提としているが、それらは同様のフロ

一チャートに従って動作し、また、それらの間のデータの相関はないので、代表として1つのプロセッサエレメント2における処理フローを表記した。なお、図5において、破線で示した矢印の部分は、その後の処理が、その前の処理で律速されるのではなく、他の処理系からの情報の入力待ちとなることを示している。

【0132】まず、ホスト計算機1の処理手順を説明する。ホスト計算機1は、係数行列の初期設定(ステップS101)を行った後、(数式18)に従って係数行列から密度行列を計算する(ステップS102)。次に、特定のプロセッサエレメント2に割り当てる縮約シェル番号RとTの組み合わせ、すなわち(RT)ペア番号を決める(ステップS103)。(RT)ペア番号の割り当て順序は任意であり、ランダムに決めても良いし、ある一定の規則に従って決めても良い。

【0133】次に、(RT)ペア番号に対応する密度行列情報を、その(RT)ペア番号が割り当てられたプロセッサエレメントに対して送信する(ステップS104)。送信される密度行列情報は、カットオフ条件を考慮して作成されるが、そのときにホスト計算機1が行う処理内容に関しては後述する。

【0134】プロセッサエレメント2は、このホスト計算機からの密度行列情報を受信し、受信バッファメモリに格納する(ステップS201)。そして、割り当てられた(RTペア)のジョブについて、縮約シェルS、Uに関するループを制御して、フォック行列要素の計算を行う(ステップS202)。そして、計算が終了すると、求めたフォック行列情報を送信バッファメモリに格納し、その送信バッファメモリからホスト計算機1に、その求めたフォック行列情報を送信する(ステップS203)。

【0135】ホスト計算機1は、以上のようにして、あるプロセッサエレメント2での処理が終了して、そのプロセッサエレメント2からフォック行列情報を受信すると(ステップS105)、そのプロセッサエレメント2に対する全てのプロセッサ・エレメントに対する(RT)ペア番号の割り当てと密度行列情報の送信が終了したかどうか判断し(ステップS107)、終了していなければ、ステップS103に戻って、新たな(RT)ペア番号の割り当てを決め、その(RT)ペア番号に対応する密度行列情報を、そのプロセッサエレメント2に送信する。

【0136】同様の操作を繰り返して、ホスト計算機1は、全てのプロセッサエレメントに対する(RT)ペア番号の割り当てと密度行列情報の送信を行い、プロセッサエレメントからのフォック行列の受信を待つ。

【0137】ステップS106で、全ての(RT)ペア番号の処理が終了したと判断すると、ホスト計算機1は、収集したフォック行列情報をもとに、図26の(数式14)を解き、新たな係数行列を計算する(ステップ

S107)。そして、新たな係数行列とその直前の係数行列とを比較して、自己無撞着な解が得られたかどうか判断する(ステップS108)。そして、両者が十分良く一致していれば、自己無撞着な解が得られたと判断し、計算を終了する。そうでなければ、ステップS108からステップS102に戻り、(数式18)に従って新たな密度行列を生成し、同様の操作を繰り返す。

【0138】ここで、ホスト計算機1におけるカットオフ処理について説明を加える。

【0139】縮約シェルRが与えられ、その原子核座標を、その縮約シェルRを構成する原始基底の軌道指数の最小値を ζ_A 、その原始基底を i とする。また、任意の縮約シェルSの原子核座標を、その縮約シェルSを構成する原始基底の軌道指数の最小値を ζ_B 、その原始基底を j とする。

【0140】従来の技術の欄で記したように、任意の原始基底 k 、1に対して、2電子積分 $g(i, j, k, l)$ をカットオフできるかどうか、(ζ_A, A, ζ_B, B)の数値の組み合わせによって判断できる。例えば、図32の(数式47)を満足するかどうかを判定し、この(数式47)の不等式が成り立っていれば、カットオフできると判断する。

【0141】それぞれ同じ縮約シェルに属する任意の2つの原始基底の組み合わせで考えると、これらの軌道指数は、上記のものよりも大きい場合、その組み合わせで計算した指数関数の数値は、上記のものよりも必ず小くなる。

【0142】したがって、この最小の軌道指数の組み合わせでカットオフできると判断された場合、縮約シェルRを構成する任意の原始基底と、縮約シェルSを構成する任意の原始基底との組み合わせで、必ずカットオフできることになる。

【0143】このように、縮約シェルSとして全ての縮約シェルを当てはめて判定を行い、縮約シェルRとのペアでのカットオフに生き残る縮約シェルの番号を1つの集合とすることができる。

【0144】ホスト計算機1は、SCF計算の開始前に、全ての縮約シェルに対して、それとのペアでのカットオフに生き残る番号の集合を、データベースとして作成しておく。これを、この実施の形態では、カットオフテーブルと呼ぶことにする。

【0145】図6に、このカットオフテーブルの一例を示す。この例において、例えば、番号が1の縮約シェルが、Rとして選ばれた($R=1$)場合、カットオフで生き残る縮約シェルSの番号は1, 2, 3, 5となる。但し、RT並列アルゴリズムで、プロセッサエレメント2において計算が行われるのは、 $S \leq R$ の場合のみであるので、 $R=1$ が割り当てられたプロセッサエレメント2で計算が行われるSの番号は、1のみとなる。

【0146】別の例として、 $R=4$ の場合を考えると、

カットオフで生き残り、しかも、 $S \leq R$ を満たすSの番号は、2, 3, 4となる。

【0147】図6のカットオフテーブルは、TUの組み合わせにおけるカットオフ判断にも全く同様に用いることができる。例えば、 $T=1$ の縮約シェルに対して、カットオフで生き残る縮約シェルUの番号は、1, 2, 3, 5である。ただし、RT並列アルゴリズムで、プロセッサエレメント2において計算が行われるのは、 $U \leq R$ の場合であるので、 $T=1$ とともに、プロセッサエレメント2に割り当てられたRの番号に依存して、プロセッサエレメント2で計算が行われるUの番号は変わる。

【0148】例えば、 $R=10$ であれば、Uとして、1, 2, 3, 5が用いられるし、 $R=2$ ならば、Uとして、1, 2が用いられる。

【0149】さて、SCF計算が開始され、あるプロセッサエレメント2に割り当てるRTペア番号が決まると、ホスト計算機1は、このカットオフテーブルを参照しながら密度行列情報を作成する。

【0150】縮約シェルR, S, T, Uを構成する縮約基底を、I, J, K, Lで、それぞれ表わすことにすると、ホスト計算機1からプロセッサエレメント2へ送信する密度行列は、 $P(I, J)$, $P(I, L)$, $P(K, L)$, $P(K, J)$ であり、また、プロセッサエレメント2で計算される2電子積分は、 $G(I, J, K, L)$ である。

【0151】 $P(I, J)$ が必要かどうかは、 $G(I, J, K, L)$ が、カットオフで生き残るかどうかに依存し、従って、カットオフテーブルに記述されている番号から、Sとして用いられる番号をリストアップし、それらの縮約シェルを構成する縮約基底をJとして $P(I, J)$ を選び出せば良い。

【0152】このように、Jに対して、 $G(I, J, K, L)$ がカットオフで生き残るかどうかは、RSの組み合わせでカットオフされないかどうかで決まるので、 $P(K, J)$ が必要かどうか全く同様に決めることができる。すなわち、 $P(I, J)$ と $P(K, J)$ とでは、送信すべきJの番号の集合は全く同じである。

【0153】したがって、 $S \leq R$ を満たすR個の縮約シェルの中で、RSの組み合わせで生き残る N_v 個の縮約シェルの集合を、 $V = \{V[1], V[2], \dots, V[N_v]\}$ とすると、 $P(I, J)$, $P(K, J)$ に係わるものとして、Vの要素の個数 N_v と、Vの要素すなわち $V[1], V[2], \dots, V[N_v]$ と、 $V[x]$ ($x=1 \sim N_v$)に係わる密度行列データブロックとを、ホスト計算機1からプロセッサエレメント2へ送信する密度行列情報に含めれば良い。

【0154】ここで、 $V[x]$ に係わる密度行列データブロックの内容は、縮約シェルRを構成する全ての縮約基底Iと縮約シェル $V[x]$ とを構成する全ての縮約基底Mの任意の組み合わせに対する $P(I, M)$ の数値、

および縮約シェルTを構成する全ての縮約基底Kと縮約シェル $V[x]$ とを構成する全ての縮約基底Mの任意の組み合わせに対する $P(K, M)$ の数値であれば良い。

【0155】全く同様に、用いられる縮約シェルUの番号を、カットオフテーブルからリストアップし、プロセッサエレメント2で用いられる $P(I, L)$, $P(K, L)$ を選び出すことが可能である。

【0156】 $U \leq R$ を満たすR個の縮約シェルの中でTUの組み合わせで生き残る N_w 個の縮約シェルの集合を、 $W = \{W[1], W[2], \dots, W[N_w]\}$ とすると、 $P(I, L)$, $P(K, L)$ に係わるものとして、Wの要素の個数 N_w と、Wの要素すなわち $W[1], W[2], \dots, W[N_w]$ と、 $W[x]$ ($x=1 \sim N_w$)に係わる密度行列データブロックとを、ホスト計算機1からプロセッサエレメント2へ送信する密度行列情報に含めれば良い。

【0157】ここで、 $W[x]$ に係わる密度行列データブロックの内容は、縮約シェルRを構成する全ての縮約基底Iと縮約シェル $W[x]$ とを構成する全ての縮約基底Nの任意の組み合わせに対する $P(I, N)$ の数値、および縮約シェルTを構成する全ての縮約基底Kと縮約シェル $W[x]$ とを構成する全ての縮約基底Nの任意の組み合わせに対する $P(K, N)$ の数値であれば良い。

【0158】〔転送データの形式〕次に、密度行列情報およびフォック行列情報の内容に関して説明する。

【0159】図7に、ホスト計算機1からプロセッサエレメント2へ送信する密度行列情報のフォーマットの一例を示す。縮約シェルRの番号から縮約シェル $W[N_w]$ の番号までは、整数型データである。また、縮約シェル $V[1]$ に関わる密度行列データブロックから縮約シェル $W[N_w]$ に関わる密度行列データブロックは、1つまたは複数の浮動小数点型データ、または、固定小数点型データにより構成されるブロックである。

【0160】図7の上から2つの縮約シェルRおよび縮約シェルTの番号により、その密度行列情報を受け取ったプロセッサエレメント2が処理すべきRTペア番号が決まる。ここで、 $T \leq R$ であるので、図32の(数式48)に示す式によって、RTペア番号(RT)を一意的に決め、2つの縮約シェル番号を用いる代わりに、1つの整数型データ(RT)によって、RTペア番号を決めることも可能である。

【0161】このようにすることで、転送すべき情報量を、整数データ1つ分だけ減らすことも可能である。しかしながら、密度行列情報全体のデータ量と比較して、整数データ1つの差は、無視できる程度に小さい上、上記のようにして決めた整数データ(RT)から縮約シェルRおよびTの番号を復元するために、プロセッサエレメント2上で行うべき処理が増加するため、図7に示すように、縮約シェルRおよびTの番号を、別々に記述する形式を用いるのが、より望ましい。

【0162】図7において、縮約シェルRおよびTの番号の次は、縮約シェルの集合Vおよび縮約シェルの集合Wの要素数を表わす2つの整数型データである。

【0163】密度行列データブロックの構成例を、縮約シェルV[1]に関わる密度行列データブロックを代表例として図8に示す。

【0164】縮約シェルの軌道量子数が1より大きい場合には、その縮約シェルに関わる密度行列データブロックは、さらに縮約基底に関わるサブブロックにより構成されることになる。図8に示した例は、縮約シェルV[1]を構成する縮約基底が、M1(V[1]), M2(V[1]), ..., Mm(V[1])と、m個ある場合であり、それぞれの縮約基底に対応したサブブロックがm個ある。

【0165】1つのサブブロックは、さらに2つの領域に別れる。1つは、縮約シェルRを構成する縮約基底I1(R)からIi(R)と、M1(V[1])とをインデックスとする密度行列要素の領域である。もう1つは、縮約シェルTを構成する縮約基底K1(T)からKk(T)と、M1(V[1])とをインデックスとする密度行列要素の領域である。ここで、i, kは、それぞれ縮約シェルR, Tを構成する縮約基底の個数である。これらの領域に含まれる密度行列要素の個数は、縮約シェルRおよびTの軌道量子数によって決まり、したがって、1組の密度行列情報の中では、どのサブブロックも同じ大きさとなる。

【0166】図9に、プロセッサエレメント2からホスト計算機1に送信するフォック行列情報のフォーマットの一例を示す。上から2つの縮約シェルRの番号および縮約シェルTの番号のみが整数型データであり、それ以降の、縮約シェルV[1]に関わるフォック行列データブロックから縮約シェルW[Nw]に関わるフォック行列データブロックは、1つまたは複数の浮動小数点型データまたは固定小数点型データにより構成されるブロックである。

【0167】密度行列情報の場合と同様に、上から2つの縮約シェルRおよび縮約シェルTの番号により、そのフォック行列情報を送信したプロセッサエレメント2が処理したRTペア番号を、ホスト計算機1が認識することができる。また、これらは、RとTから一意に決まる1つの整数型データであるRTペア番号で代用することも可能である。

【0168】さらに、また、ホスト計算機1上に、プロセッサエレメント2に割り当てたRTペア番号を記録しておき、RTペアを表わす番号の代わりに、プロセッサエレメント認識番号により、プロセッサエレメント2が送信したフォック行列情報が、どのRTペアに対応するものかを、ホスト計算機1に判断させることも可能である。

【0169】フォック行列データブロックの構成は、図

8に示した密度行列データブロックと同様である。なお、フォック行列情報には、密度行列情報を構成している、縮約シェルの集合VおよびWの要素数や縮約シェルの番号情報を含む必要がない。なぜならば、ホスト計算機1上にはカットオフテーブルがあるため、ホスト計算機1は、RおよびTから、これらの情報を簡単に再生することが可能だからである。

【0170】ホスト計算機1上に十分なデータ保持容量がある場合には、プロセッサエレメント2へ密度行列情報を送信してから、プロセッサエレメント2からフォック行列情報を受信するまでの間、これらの情報を保持しておくことができる。いずれにしても、プロセッサエレメント2からホスト計算機1へ送信するフォック行列情報に、縮約シェルの集合VおよびWの要素数や縮約シェルの番号情報を含む必要はなく、これらをフォック行列情報に含まないことによって、プロセッサエレメント2とホスト計算機1との間の通信データ量を小さくすることができる。

【0171】図10に、密度行列情報およびフォック行列情報のプロセッサエレメント2内のメモリ空間への割付の一例を示す。

【0172】図10では、プロセッサエレメント2のメモリ空間へのアドレッシングが、全てワード・アドレッシングで行われることを前提としている。また、縮約シェルの集合Vの要素数を1000、Wの要素数を500とした場合を示している。

【0173】0番地および1番地に、そのプロセッサエレメント2に割り当てられたRおよびTの番号を格納する。2番地および3番地には、縮約シェルの集合VおよびWの要素数Nv, Nwを格納する。4番地から(4+Nv-1)番地は、縮約シェルV[1]からV[Nv]の番号を格納する領域、(4+Nv)番地から(4+Nv+Nw-1)番地は、縮約シェルW[1]からW[Nw]の番号を格納する領域である。以上には全て整数型のデータが保存される。

【0174】空き領域を挟んで、2000番地から9999番地は、縮約シェルV[1]からV[Nv]に係わる密度行列データブロックを格納する領域、10000番地から17999番地は、縮約シェルW[1]からW[Nw]に係わる密度行列データブロックを格納する領域である。これらの領域に、P(I, J), P(K, J), P(I, L), P(K, L)を、それぞれ4000個ずつ格納することが可能である。

【0175】18000番地から25999番地は、縮約シェルV[1]からV[Nv]に係わるフォック行列データブロックを格納する領域、26000番地から33999番地は、縮約シェルW[1]からW[Nw]に係わるフォック行列データブロックを格納する領域である。これらの領域に、F(I, J), F(K, J), F(I, L), F(K, L)を、それぞれ4000個ずつ

格納することが可能である。

【0176】メモリ上で、1ワードが64ビットであると仮定しても、これらのデータ領域の容量は、せいぜい2Mビット程度である。カットオフで生き残るデータのみが格納されることを考慮すると、数1000基底以上の規模の分子軌道計算に対しても、十分に必要なデータを格納する領域がある。なお、この例よりも大きなデータ保持能力を有するメモリを実装することも、現在の技術では非常に安価に実現できる。

【0177】[ループの回しかた(プロセッサエレメント2の処理手順)]次に、プロセッサエレメント2における処理の手順を、図11および図12のフローチャートに従って説明する。

【0178】まず、プロセッサエレメント2は、ホスト計算機1から密度行列情報を受信し、それをメモリに格納する(ステップS301)。格納が済んだ時点から、図5に示したSおよびUに関するループによる計算を開始する。

【0179】Sに関するループは、以下のように制御される。まず、整数型変数 v を用意し、初期値をゼロにセットする(ステップS302)。次に、ループに入り、変数 v を1だけ増加させる(ステップS303)。次に、メモリから縮約シェル $V[v]$ の数値を読み出し、それをSに代入する(ステップS304)。

【0180】以下、適当な処理を終了すると、変数 v と、縮約シェル V の個数 N_v とを比較する(図12のステップS323)。変数 v と N_v とが等しければ、Sに関するループを終了し、フォック行列情報をホスト計算機1へ送信して(ステップS324)、割り当てられたRTペアの処理を終了する。その後、プロセッサエレメント2は密度行列情報の受信待ちの状態となる。変数 v と N_v とが等しくなければ、ステップS303で変数 v を1だけ増加させて、同様の処理を繰り返す。

【0181】同様に、Uに関するループは以下のように制御される。Sに関するループ内で、整数型変数 w を用意し、初期値をゼロにセットする(ステップS305)。次に、ループに入り、変数 w を1だけ増加させる(ステップS306)。次に、メモリから縮約シェル $W[w]$ の数値を読み出し、それをUに代入する(ステップS307)。

【0182】以下、適当な処理を終了すると、変数 w と縮約シェル W の個数 N_w とを比較する(ステップS322)。変数 w と N_w とが等しければ、Uに関するループを終了し、前記ステップS323の変数 v と N_v とが等しいかどうかの判断を行う。ステップS322で、変数 w と N_w とが等しくなければ、ステップS306に戻って、変数 w を1だけ増加させて、同様の処理を繰り返す。

【0183】このように、SおよびUに関するループは、メモリに格納された縮約シェル $V[1]$ 、 V

$[2]$ 、 \dots 、 $V[N_v]$ や、縮約シェル $W[1]$ 、 $W[2]$ 、 \dots 、 $W[N_w]$ の数値を、順次、読み出して、それらをSおよびUの数値として用いることにより制御され、その結果、プロセッサエレメント2は、カットオフで生き残るSおよびUのみに関する数値にしかセットされないため、非常に効率よくループ制御を行うことができる。

【0184】分子軌道計算における2電子積分およびフォック行列要素の計算は、縮約シェル単位でなく、縮約基底を単位として行う。そのため、Uに関するループの内側では、縮約シェルRを構成する縮約基底Iに関する $b_basis(R)$ から $e_basis(R)$ までのループ(ステップS308、ステップS309およびステップS321参照)、縮約シェルSを構成する縮約基底Jに関する $b_basis(S)$ から $e_basis(S)$ までのループ(ステップS310、ステップS311およびステップS320参照)、縮約シェルTを構成する縮約基底Kに関する $b_basis(T)$ から $e_basis(T)$ までのループ(ステップS312、ステップS313およびステップS319参照)、縮約シェルUを構成する縮約基底Lに関する $b_basis(U)$ から $e_basis(U)$ までのループ(ステップS314、ステップS315およびステップS318参照)よりなる4重ループを形成し、その中で、2電子積分 $G(I, J, K, L)$ の計算(ステップS316)、および2電子積分と密度行列要素との積をフォック行列要素に足し込む計算(ステップS317)を行う。

【0185】ここで、同一の縮約シェルを構成している縮約基底の番号が、連続となるように縮約基底の番号付けがなされていることを前提として、縮約シェルXを構成する縮約基底の番号が、 $b_basis(X)$ から $e_basis(X)$ の範囲にあるものとした。

【0186】この4重ループの中では、カットオフとして上記のRSおよびTUの組み合わせによるもの以外を考えない限りは、必ず、2電子積分 $G(I, J, K, L)$ の計算が行われる。

【0187】したがって、SおよびUのループがまわる間に、メモリに格納された密度行列要素の全てが、必ず、フォック行列要素への足し込み計算に用いられる。計算で用いられないような密度行列要素の通信を行うことは、通信量を無意味に増加させることになるが、そのような無駄な通信が生じないようにしていることにより、低価格な通信手段を用いた場合においてさえ、システムの処理性能が通信性能により制限されることがなく、システムの処理効率を向上させることが可能である。

【0188】[データ転送のタイミングおよび行列がメモリに入りきらない場合の方法]縮約シェルSおよびUに関するループのうち、外側にあるSに係わる密度行列

10

20

30

40

50

要素 $P(I, J)$, $P(K, J)$ やフォック行列要素 $F(I, J)$, $F(K, J)$ は、2 電子積分 $G(I, J, K, L)$ の計算終了後に、その都度必要な密度行列要素を、ホスト計算機から受信し、計算し終わったフォック行列要素を、その都度ホスト計算機へ送信する、という手順で送受信することも可能である。

【0189】しかしながら、データの通信時には、純粹に通信するデータだけでなく通信プロトコルに係る付加的な情報が必ず付随するため、通信する情報の単位を細かくすることにより、通信量を増大させることに繋がる。

【0190】この実施の形態のように、RT ペアにつき、密度行列情報およびフォック行列情報を、それぞれ 1 回の通信で、ホスト計算機とプロセッサ・エレメントとの間で通信することにより、プロトコルに付随する付加的な情報による通信量の増大を最小限にとどめることができる。

【0191】なお、計算の対象とする系のサイズが大きかったり、カットオフ判断を行うための閾値が大きく、カットオフによる生き残り割合が高い場合には、密度行列要素 $P(I, J)$, $P(K, J)$ 、フォック行列要素 $F(I, J)$, $F(K, J)$ 、および密度行列要素 $P(I, L)$, $P(K, L)$ 、フォック行列要素 $F(I, L)$, $F(K, L)$ の全てをプロセッサエレメント 2 内のメモリに保存できなくなることがある。その場合の処理方法を説明する。

【0192】まず、密度行列要素 $P(I, J)$, $P(K, J)$ と、フォック行列要素 $F(I, J)$, $F(K, J)$ とが、全てプロセッサエレメント 2 内のメモリに保存可能である場合には、これら全てと、密度行列要素 $P(I, L)$, $P(K, L)$ およびフォック行列要素 $F(I, L)$, $F(K, L)$ がメモリに保存できるように、縮約シェル U の番号を区切る。

【0193】このことにより、(RT) ペアで指定されるジョブは、複数のサブジョブに分割される。これらのサブジョブは、同一のプロセッサエレメント 2 に割り当てられるようにした上で、密度行列要素 $P(I, J)$, $P(K, J)$ は、最初のサブジョブの開始前に、1 度だけホスト計算機 1 からプロセッサエレメント 2 に送信する。

【0194】これらの密度行列要素は、縮約シェル U の値が、いくらであろうとも、全て共通に用いることができるからである。一方、密度行列要素 $P(I, L)$, $P(K, L)$ は、各サブジョブの開始前に、ホスト計算機 1 からプロセッサエレメント 2 に送信し、その前のサブジョブで用いられたメモリ上の密度行列要素 $P(I, L)$, $P(K, L)$ の領域に上書きしてしまつて構わない。これらの密度行列要素は、特定の U に係わる計算にしか用いられないからである。

【0195】同様に、フォック行列要素 $F(I, L)$,

$F(K, L)$ を、各サブジョブの終了後に、プロセッサエレメント 2 からホスト計算機 1 に送信し、そのメモリ領域を、次のサブジョブで計算されるフォック行列要素 $F(I, L)$, $F(K, L)$ の格納領域として開放する。

【0196】一方、フォック行列要素 $F(I, J)$, $F(K, J)$ は、全てのサブジョブで共通に用いることができるので、全てのサブジョブの終了後に、1 度だけプロセッサエレメント 2 からホスト計算機 1 へ送信する。

【0197】密度行列要素 $P(I, J)$, $P(K, J)$ とフォック行列要素 $F(I, J)$, $F(K, J)$ との全ては、プロセッサエレメント 2 内のメモリに保存することができないが、密度行列要素 $P(I, L)$, $P(K, L)$ と、フォック行列要素 $F(I, L)$, $F(K, L)$ との全てを、プロセッサエレメント 2 内のメモリに保存可能な場合には、縮約シェル S の番号を区切って、ジョブを複数のサブジョブに分割する。

【0198】このような処理方法とすることにより、全ての行列要素を、プロセッサエレメント 2 内のメモリに保存可能な場合と比較した通信量の増加分は、密度行列要素とフォック行列要素とを分割して送受信すること起因する通信プロトコルに係る付加的な情報の個数の増分のみとなる。

【0199】密度行列要素 $P(I, J)$, $P(K, J)$ とフォック行列要素 $F(I, J)$, $F(K, J)$ との全て、および密度行列要素 $P(I, L)$, $P(K, L)$ とフォック行列要素 $F(I, L)$, $F(K, L)$ との全てのどちらも、プロセッサエレメント 2 内のメモリに格納できないような、計算規模が非常に大きい場合に対しても、この実施の形態の RT 並列アルゴリズムは対応可能である。以下に、その処理アルゴリズムを説明する。

【0200】密度行列要素 $P(I, J)$, $P(K, J)$ とフォック行列要素 $F(I, J)$, $F(K, J)$ (以下では行列情報 A とする) との全てで、データサイズが X であり、密度行列要素 $P(I, L)$, $P(K, L)$ とフォック行列要素 $F(I, L)$, $F(K, L)$ (以下では行列情報 B とする) との全てで、データサイズが Y であると仮定する。

【0201】また、縮約シェル番号 S を区切って、行列情報 A を x 等分し、縮約シェル番号 U を区切って、行列情報 B を y 等分したときに、 $X/x + Y/y$ のサイズのデータ量であれば、プロセッサエレメント 2 内のメモリに保存可能であると仮定する。(RT) ペアで指定される 1 つのジョブが、 $x \times y$ 個のサブジョブに分割されたことになる。

【0202】図 13 に示すように、サブジョブの番号を (S に関する分割番号、U に関する分割番号) の形式で表わし、(1, 1) から (1, y) をプロセッサエレメント PE1 に、(2, 1) から (2, y) をプロセッサエレメント PE2 に、(x, 1) から (x, y) をプロ

セッサエレメント PEx に、というようにプロセッサエレメントへの割り付けを行う（第 1 の分割方法）ことにすると、1 つのプロセッサエレメントに係わって、ホスト計算機 1 との間で送受信されるデータ量は、 $X/x + y \times Y/y$ となる。

【0203】このような通信を、x 個のプロセッサエレメントがそれぞれ行うので、(RT) ペアにより指定される 1 つのジョブ当たりで、

$$x \times [X/x + y \times Y/y] = X + xY$$

の通信量となる。

【0204】上記の例とは逆に、(1, 1) から (x, 1) をプロセッサエレメント PE1 に、(1, 2) から (x, 2) をプロセッサエレメント PE2 に、(1, y) から (x, y) をプロセッサエレメント PEy に、というように、プロセッサエレメントへの割り付けを行う（第 2 の分割方法）場合、(RT) ペアにより指定される 1 つのジョブ当たりの通信量は、

$$yX + Y$$

となる。

【0205】プロセッサエレメント 2 内のメモリ容量は限られているので、縮約シェル番号の区切り数を少なくするためには元々のデータ量になるべく小さい方の区切りで別々のプロセッサエレメント 2 にサブジョブを割り当てるのが良い。

【0206】第 1 の分割方法と第 2 の分割方法とで大きさを比較してみる。x は X に比例し、y は Y に比例して、その比例定数は近似的に共通とすることができるので、第 1 の分割方法と第 2 の分割方法とでの通信量の差は、 $X - Y$ となる。したがって通信量は、 $X > Y$ の場合には、第 2 の分割方法の方が小さく、 $X < Y$ の場合には、第 1 の分割方法の方が小さくなる。

【0207】すなわち、密度行列要素 P(I, J), P(K, J) とフォック行列要素 F(I, J), F(K, J) の要素数の和が、密度行列要素 P(I, L), P(K, L) とフォック行列要素 F(I, L), F(K, L) の要素数の和よりも大きい ($X > Y$) 場合には、第 2 の分割方法を用いる。逆に、密度行列要素 P(I, J), P(K, J) とフォック行列要素 F(I, J), F(K, J) の要素数の和が、密度行列要素 P(I, L), P(K, L) とフォック行列要素 F(I, L), F(K, L) の要素数の和よりも小さい ($X < Y$) 場合には、第 1 の分割方法を用いることにより、通信量の増大分を小さくすることが可能である。

【0208】図 14 に、ジョブの分割を考慮した場合のホスト計算機における処理のアルゴリズムを示す。この図 14 は、図 5 中のホスト計算機の処理アルゴリズムにおけるステップ S103 からステップ S106 までの部分に対応するもので、図 14 のステップ S401 が図 5 のステップ S103 に、図 14 のステップ S411 が図 5 のステップ S106 に、それぞれ対応している。な

お、この図 14 では、プロセッサエレメントは、簡単のため PE と記述した。

【0209】すなわち、図 14 において、ステップ S401 では、プロセッサエレメント 2 に割り当てる (RT) ペア番号を決定する。その後、ホスト計算機 1 は、まず、(RT) ペア番号が割り当てられたプロセッサエレメント 2 のメモリに、P(K, L), P(I, L), F(K, L), F(I, L) の全てを格納可能かどうかを判断する（ステップ S402）。

10 【0210】ステップ S402 で、格納可能であると判断されたときには、次に、(RT) ペア番号が割り当てられたプロセッサエレメント 2 のメモリに、P(I, J), P(K, J), F(I, J), F(K, J) の全てを格納可能かどうかを判断し（ステップ S405）、格納可能であれば、ステップ S406、ステップ S407 に順に進む。また、格納可能でなければ、後述する図 15 に示す処理に進む。

【0211】ステップ S406、407 の処理は、プロセッサエレメント 2 に割り当てられた (RT) ペア番号に相当する処理に必要な密度行列およびフォック行列を、全て、プロセッサエレメント 2 のメモリに格納可能な場合の処理であるので、図 5 における処理と同様である。すなわち、P(K, L), P(I, L), P(I, J), P(K, J) の全てをプロセッサエレメント 2 へ送信する（ステップ S406）。その後は、処理が終了して、プロセッサエレメント 2 から、F(K, L), F(I, L), F(I, J), F(K, J) の全てを受信するまで、ホスト計算機は処理待ちの状態となる（ステップ S407）。

30 【0212】また、ステップ S402 で、格納可能でないと判断されたときには、(RT) ペア番号が割り当てられたプロセッサエレメント 2 のメモリに、P(I, J), P(K, J), F(I, J), F(K, J) の全てが格納可能かどうかを判断し（ステップ S403）、格納可能であると判断されたときには、後述する図 16 に示す処理を行い、そうでなければステップ S404 に進む。

【0213】ステップ S404 では、P(I, J) と P(K, J) の個数の和と、P(K, L) と P(I, L) の個数の和とを比較し、等しいかあるいは後者が大きければ、後述する図 17 に示す処理を行ない、前者が大きければ、後述する図 18 に示す処理を行なう。

【0214】図 15、図 16、図 17 および図 18 の処理のいずれかの処理が終了すると、ホスト計算機 1 は、全ての (RT) ペアの処理が終了したかどうかを判断する（ステップ S411）。以下、図 5 の処理と同様である。

【0215】次に、図 15 に示す処理を説明する。この処理は、プロセッサエレメント 2 に割り当てられた (RT) ペア番号に相当する処理に必要な密度行列およびフ

フォック行列の全ては、プロセッサエレメント2のメモリに格納できないが、 $P(K, L)$, $P(I, L)$, $F(K, L)$, $F(I, L)$ の全てを格納することができる場合の処理である。

【0216】まず、 $P(K, L)$, $P(I, L)$, $F(K, L)$, $F(I, L)$ の全てを格納した場合に、プロセッサエレメント2のメモリに残る空き容量を見積り、対応する $P(I, J)$, $P(K, J)$, $F(I, J)$, $F(K, J)$ の全てがその空き容量以下となるように、 S の範囲を M 分割する。それとともに、分割された各 S の範囲を単位として M 個のサブジョブを定義する(ステップS408)。

【0217】次に、 $P(K, L)$, $P(I, L)$ の全てをプロセッサエレメント2に送信する(ステップS409)。その後、 $m=1$ から $m=M$ までに対して、 m 番目のサブジョブに対応する $P(I, J)$, $P(K, J)$ の全てをプロセッサエレメント2へ送信し、処理待ちの状態を経て、プロセッサエレメント2から m 番目のサブジョブに対応する $F(I, J)$, $F(K, J)$ の全てを受信する、という処理を繰り返す(ステップS410～ステップS414)。

【0218】そして、 $m=1$ から $m=M$ までの繰り返しの処理の全てを終了すると、プロセッサエレメントから $F(K, L)$, $F(I, L)$ の全てを受信する(ステップS415)。その後、図14のステップS411に進む。

【0219】次に、図16に示す処理を説明する。これは、プロセッサエレメント2に割り当てられた(RT)ペア番号に相当する処理に必要な密度行列およびフォック行列の全てをプロセッサエレメント2のメモリに格納できず、また、 $P(K, L)$, $P(I, L)$, $F(K, L)$, $F(I, L)$ の全てを格納することもできないが、 $P(I, J)$, $P(K, J)$, $F(I, J)$, $F(K, J)$ の全てを格納することができる場合の処理である。

【0220】まず、 $P(I, J)$, $P(K, J)$, $F(I, J)$, $F(K, J)$ の全てを格納した場合に、プロセッサエレメント2のメモリに残る空き容量を見積り、対応する $P(K, L)$, $P(I, L)$, $F(K, L)$, $F(I, L)$ の全てがその空き容量以下となるように U の範囲を M 分割する。それとともに、分割された各 U の範囲を単位として M 個のサブジョブを定義する(ステップS416)。

【0221】次に、 $P(I, J)$, $P(K, J)$ の全てをプロセッサエレメント2に送信する(ステップS417)。その後、 $m=1$ から $m=M$ までに対して、 m 番目のサブジョブに対応する $P(K, L)$, $P(I, L)$ の全てをプロセッサエレメントへ送信し、処理待ちの状態を経て、プロセッサエレメントから m 番目のサブジョブに対応する $F(K, L)$, $F(I, L)$ の全てを受信す

る、という処理を繰り返す(ステップS418～ステップS422)。

【0222】そして、 $m=1$ から $m=M$ までの繰り返しの処理の全てを終了すると、プロセッサエレメント2から $F(I, J)$, $F(K, J)$ の全てを受信する(ステップS423)。その後、図14のステップS411に進む。

【0223】次に、図17に示す処理を説明する。これは、プロセッサエレメント2のメモリに、 $P(K, L)$, $P(I, L)$, $F(K, L)$, $F(I, L)$ の全てを格納することができず、また、 $P(I, J)$, $P(K, J)$, $F(I, J)$, $F(K, J)$ の全てを格納することもできない場合で、 $P(I, J)$ と $P(K, J)$ の個数の和が $P(K, L)$ の $P(I, L)$ 個数の和よりも大きくない場合の処理である。

【0224】まず、 $P(I, J)$, $P(K, J)$, $F(I, J)$, $F(K, J)$ の全てがメモリの空き容量以下となるように、 S の範囲を $M1$ 分割する。 $M1$ はできるだけ小さく設定する(ステップS424)。

【0225】次に、ステップS425で、 $m1=0$ として初期化した後、ステップS426以降に進み、 $m1$ に関するループを、 $m1=1$ から $m1=M1$ の範囲で回す。 $m1$ に関するループの中では、まず、分割された $P(I, J)$, $P(K, J)$, $F(I, J)$, $F(K, J)$ の全てを格納した場合に、プロセッサエレメント2のメモリに残る空き容量を見積り、対応する $P(K, L)$, $P(I, L)$, $F(K, L)$, $F(I, L)$ の全てがその空き容量以下となるように U の範囲を $M2$ 分割する。さらに、分割された各 U の範囲を単位として $M2$ 個のサブジョブを定義する(ステップS427)。そして、次に、 $m1$ 番目の分割に対応する $P(I, J)$, $P(K, J)$ の全てを、プロセッサエレメント2へ送信する(ステップS428)。

【0226】その後、ステップS429で、 $m2=0$ として初期化した後、ステップS430以降に進み、 $m2$ に関するループを、 $m2=1$ から $m2=M2$ の範囲で回す。 $m2$ に関するループでは、まず、 $m2$ 番目の分割に対応する $P(K, L)$, $P(I, L)$ の全てをプロセッサエレメントへ送信し(ステップS431)、処理待ちの状態を経て、プロセッサエレメントから $m2$ 番目の分割に対応する $F(K, L)$, $F(I, L)$ の全てを受信する(ステップS432)。

【0227】 $m2$ に関するループが終了したと判断すると(ステップS433)、プロセッサエレメント2から $m1$ 番目の分割に対応する $F(I, J)$, $F(K, J)$ の全てを受信する(ステップS434)。そして、 $m1$ に関するループが終了したと判断すると(ステップS435)、図14のステップS411に進む。

【0228】なお、 $m1$ 番目の分割に対応するループ内の全てのサブジョブの処理は、同一のプロセッサエレ

10

20

30

40

50

ント2で行なう必要があるが、M1個に分割された異なるSの範囲に対応する処理は、それぞれを独立のジョブと見なして、別々のプロセッサエレメントで処理を行なってもよい。それらのジョブを、同一のプロセッサエレメントで行なっても、別々のプロセッサエレメントで行なっても、発生する通信量は同じである。

【0229】次に、図18に示す処理を説明する。この処理は、プロセッサエレメント2のメモリに、P(K, L), P(I, L), F(K, L), F(I, L)の全てを格納することができず、また、P(I, J), P(K, J), F(I, J), F(K, J)の全てを格納することもできない場合で、P(I, J)とP(K, J)の個数の和が、P(K, L)のP(I, L)個数の和よりも大きい場合の処理である。

【0230】まず、P(K, L), P(I, L), F(K, L), F(I, L)の全てがメモリの空き容量以下となるように、Uの範囲をM1分割する。M1はできるだけ小さく設定する(ステップS436)。次に、ステップS437で、m1=0として初期化した後、ステップS438以降に進み、m1に関するループを、m1=1からm1=M1の範囲で回す。

【0231】m1に関するループの中では、まず、分割されたP(K, L), P(I, L), F(K, L), F(I, L)の全てを格納した場合に、プロセッサエレメントのメモリに残る空き容量を見積り、対応するP(I, J), P(K, J), F(I, J), F(K, J)の全てがその空き容量以下となるように、Sの範囲をM2分割する。さらに、分割された各Sの範囲を単位としてM2個のサブジョブを定義する(ステップS439)。

【0232】m1番目の分割に対応するP(K, L), P(I, L)の全てをプロセッサエレメント2へ送信した後、ステップS441で、m2=0として初期化した後、ステップS442以降に進み、m2に関するループを、m2=1からm2=M2の範囲で回す。

【0233】m2に関するループでは、まず、m2番目の分割に対応するP(I, J), P(K, J)の全てをプロセッサエレメント2へ送信し(ステップS443)、処理待ちの状態を経て、プロセッサエレメント2からm2番目の分割に対応するF(I, J), F(K, J)の全てを受信する(ステップS444)。

【0234】m2に関するループが終了したと判断すると(ステップS445)、プロセッサエレメント2からm1番目の分割に対応するF(K, L), F(I, L)の全てを受信する(ステップS446)。そして、m1に関するループが終了したと判断すると(ステップS447)、図14のステップS411に進む。

【0235】なお、この場合も、m1番目の分割に対応するループ内の全てのサブジョブの処理は、同一のプロセッサエレメント2で行なう必要があるが、M1個に分

割された異なるUの範囲に対応する処理は、それぞれを独立のジョブと見なして、別々のプロセッサエレメント2で処理を行なってもよい。

【0236】以上説明した実施の形態は、非経験的分子軌道法を用いた分子シミュレーションにおいて、フォック行列要素計算を高速に行う場合に、この発明を適用した場合であるが、この発明は、このような非経験的分子軌道法に限らず、種々の並列処理アルゴリズムに適用可能であることは、言うまでもない。

【0237】

【発明の効果】以上説明したように、この発明によれば、安価な通信手段と比較的小さなメモリを持った並列処理システムを用いて、大規模な行列要素の計算を高速に行うことが可能となる。

【図面の簡単な説明】

【図1】この発明による並列処理装置の実施の形態のシステム構成を示すブロック図である。

【図2】この発明の実施の形態の比較例として示す従来例のFosterの列ブロックアルゴリズムでトリプルソート法のアルゴリズムを示すプログラム・コードを示す図である。

【図3】この発明の実施の形態のRT並列アルゴリズムを示すプログラム・コードを示す図である。

【図4】この発明の実施の形態のRT並列アルゴリズムにおいて、必要な通信性能と縮約シェル番号Rとの関係を示す図である。

【図5】この発明の実施の形態におけるホスト計算機およびプロセッサエレメントの処理のフローチャートである。

【図6】この発明の実施の形態におけるホスト計算機上に用意されるカットオフテーブルの一例を示す図である。

【図7】この発明の実施の形態において、ホスト計算機からプロセッサエレメントへ送信される密度行列情報のフォーマットの一例を示す図である。

【図8】この発明の実施の形態における密度行列データブロックの構成例を示す図である。

【図9】この発明の実施の形態において、プロセッサエレメントからホスト計算機へ送信されるフォック行列情報のフォーマットの一例を示す図である。

【図10】この発明の実施の形態におけるメモリへの行列情報の割り付け例を示す図である。

【図11】この発明の実施の形態におけるプロセッサエレメントの処理のフローチャートの一部を示す図である。

【図12】この発明の実施の形態におけるプロセッサエレメントの処理のフローチャートの一部を示す図である。

【図13】この発明の実施の形態における、計算規模が大きい場合のサブジョブへの分割と、それらのプロセッ

サエレメントへの割り当て方法の一例を示す図である。

【図 14】この発明の実施の形態における、ジョブ分割を考慮した場合のホスト計算機の処理アルゴリズムの一例を示すフローチャートの一部を示す図である。

【図 15】この発明の実施の形態における、ジョブ分割を考慮した場合のホスト計算機の処理アルゴリズムの一例を示すフローチャートの一部を示す図である。

【図 16】この発明の実施の形態における、ジョブ分割を考慮した場合のホスト計算機の処理アルゴリズムの一例を示すフローチャートの一部を示す図である。

【図 17】この発明の実施の形態における、ジョブ分割を考慮した場合のホスト計算機の処理アルゴリズムの一例を示すフローチャートの一部を示す図である。

【図 18】この発明の実施の形態における、ジョブ分割を考慮した場合のホスト計算機の処理アルゴリズムの一例を示すフローチャートの一部を示す図である。

【図 19】原始基底関数と、その角運動量、軌道指数、原子核座標との対応例を示す図である。

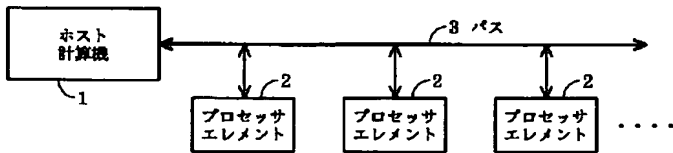
【図 20】従来例の F o s t e r の列ブロックアルゴリズムのフローチャートである。

【図 21】従来例の F o s t e r の列ブロックアルゴリズムでカノニカル法の場合における、必要な通信性能のブロックサイズ依存性を説明するための図である。

【図 22】従来例の F o s t e r の列ブロックアルゴリズムでトリプルソート法の場合における、必要な通信性能のブロックサイズ依存性を説明するための図である。

【図 23】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図 1】



【図 6】

カットオフテーブルの一例

縮約シェルXの番号	縮約シェルXとのペアでのカットオフで生き残る縮約シェル番号
1	1, 2, 3, 5
2	2, 4
3	1, 3, 5,
4	2, 3, 4, 6
5	1, 3, 5, 6, 7
⋮	⋮
100	87, 91, 92, 96, 97, 98, 99, 100

【図 24】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図 25】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図 26】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図 27】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図 28】文献 1 の 2 電子積分計算手法の説明に用いる数式を示す図である。

【図 29】文献 1 の 2 電子積分計算手法の説明に用いる数式を示す図である。

【図 30】従来例の F o s t e r のアルゴリズムの説明に用いる数式を示す図である。

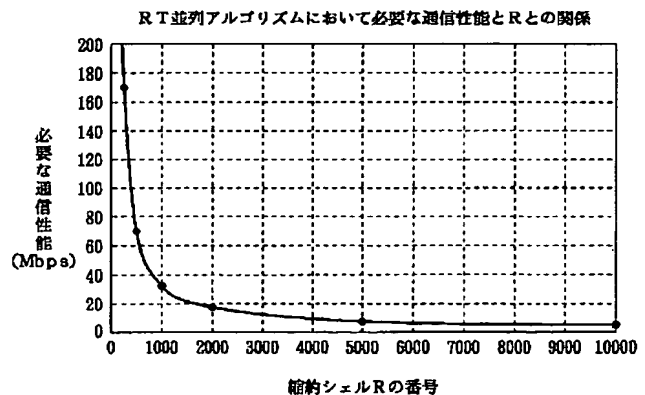
【図 31】従来例の F o s t e r のアルゴリズムの説明に用いる数式を示す図である。

【図 32】従来例における通信量（数式 4 4）、従来例における必要な通信性能（数式 4 5）、この発明の実施の形態における必要な通信性能（数式 4 6）、この発明の実施の形態におけるカットオフの判定基準の一例（数式 4 7）、この発明の実施の形態における R T ペア番号の一意的な決め方の一例（数式 4 8）に、それぞれ用いる数式を示す図である。

【符号の説明】

- 1 ホスト計算機
- 2 プロセッサエレメント
- 3 バス

【図 4】



【図2】

トリプルソートアルゴリズム (比較例)

```

for(R=1; R<=Nshell; R++){
  for(S=1; S<=R; S++){
    for(T=1; T<=S; T++){
      for(U=1; U<=T; U++){
        for(I=b_basis(R); I<=e_basis(R); I++){
          for(J=b_basis(S); J<=e_basis(S); J++){
            for(K=b_basis(T); K<=e_basis(T); K++){
              for(L=b_basis(U); L<=e_basis(U); L++){
                G_IJKL=G(I, J, K, L);
                G_IKJL=G(I, K, J, L);
                G_ILJK=G(I, L, J, K);
                F[I][J]+P[K][L]*(G_IJKL+G_IKJL+G_ILJK);
                F[I][K]-P[J][L]*(G_IJKL+G_IKJL+G_ILJK)/2;
                F[I][L]-P[J][K]*(G_IJKL+G_IKJL+G_ILJK)/2;
                F[J][K]-P[I][L]*(G_IJKL+G_IKJL+G_ILJK)/2;
                F[J][L]-P[I][K]*(G_IJKL+G_IKJL+G_ILJK)/2;
                F[K][L]+P[I][J]*(G_IJKL+G_IKJL+G_ILJK);
              }}}
            }}}
          }}}
        }}}
      }}}
    }}}
  }}}
}

```

【図3】

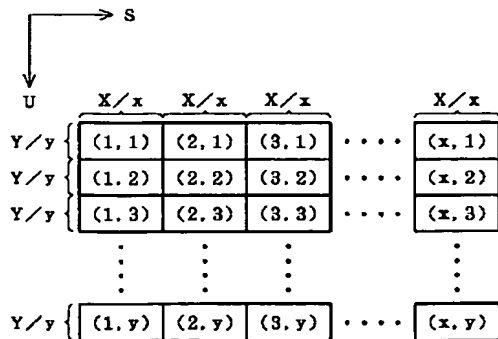
RT並列アルゴリズム (この発明の実施形態)

```

for(RT=1; RT<=Nshell*(Nshell+1)/2; RT++){
  for(S=1; S<=R; S++){
    for(U=1; U<=R; U++){
      for(I=b_basis(R); I<=e_basis(R); I++){
        for(J=b_basis(S); J<=e_basis(S); J++){
          for(K=b_basis(T); K<=e_basis(T); K++){
            for(L=b_basis(U); L<=e_basis(U); L++){
              G_IJKL=G(I, J, K, L);
              F[I][J]+P[K][L]*G_IJKL;
              if(L<=K&&K<I){
                if(J<I) F[K][L]+2*P[I][J]*G_IJKL;
                else F[K][L]+P[I][J]*G_IJKL;
              }
              F[I][L]-0.5*P[K][J]*G_IJKL;
              if(J<I&&K<=J)
                F[K][J]-0.5*P[I][L]*G_IJKL;
              if(K<I&&J<=K&&L<I)
                F[K][J]-0.5*P[I][L]*G_IJKL;
            }}}
          }}}
        }}}
      }}}
    }}}
  }}}
}

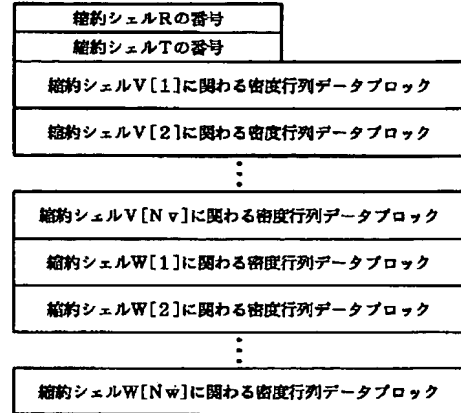
```

【図13】

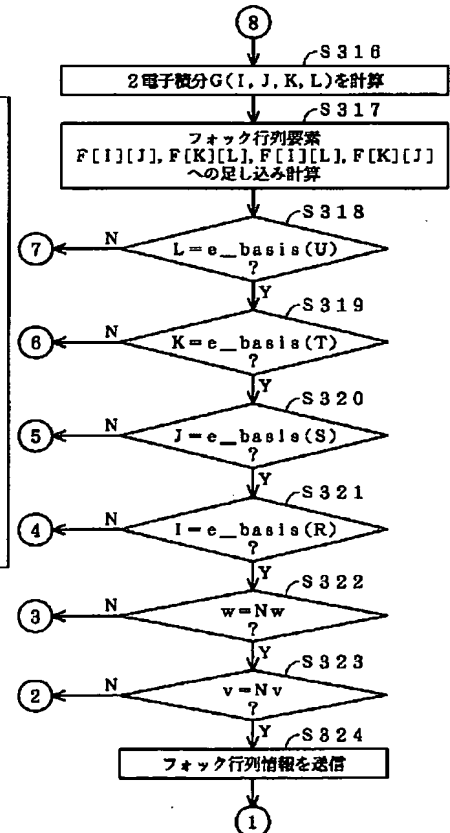


【図9】

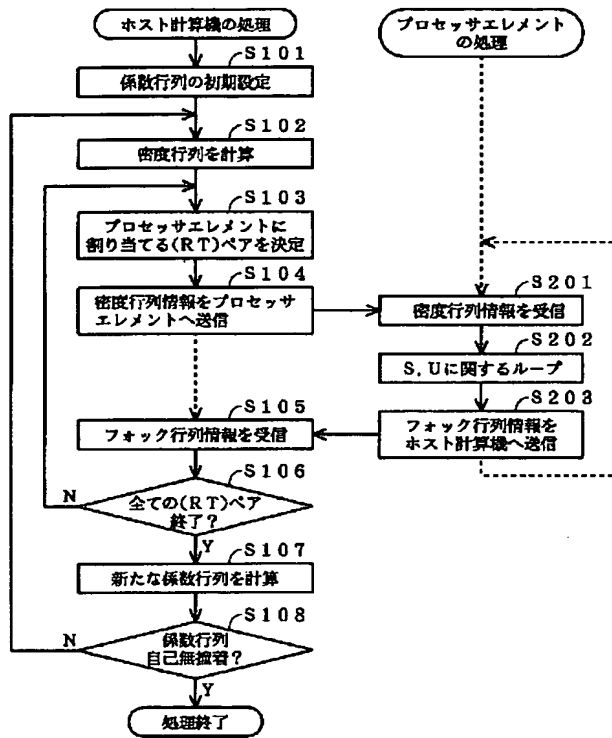
転送するフォック行列情報のフォーマット例



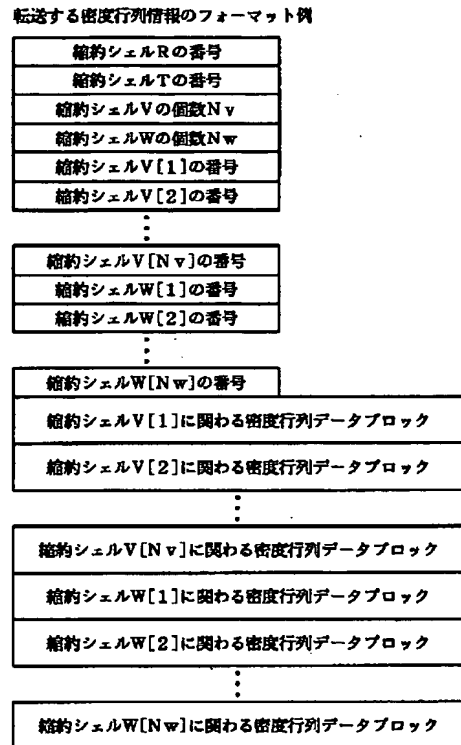
【図12】



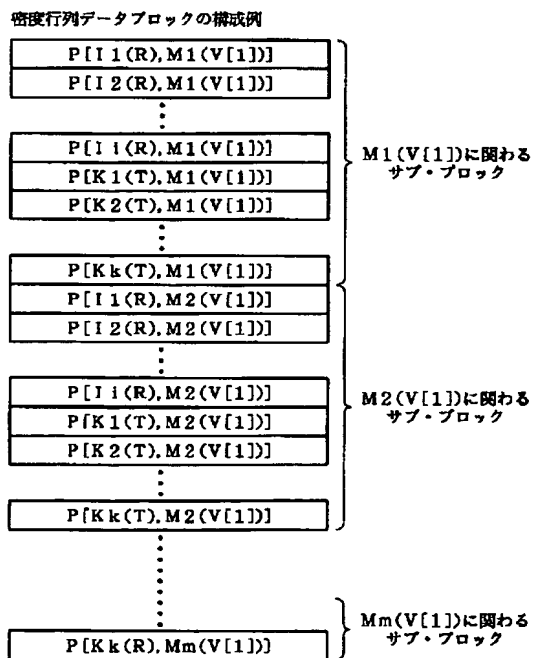
【図 5】



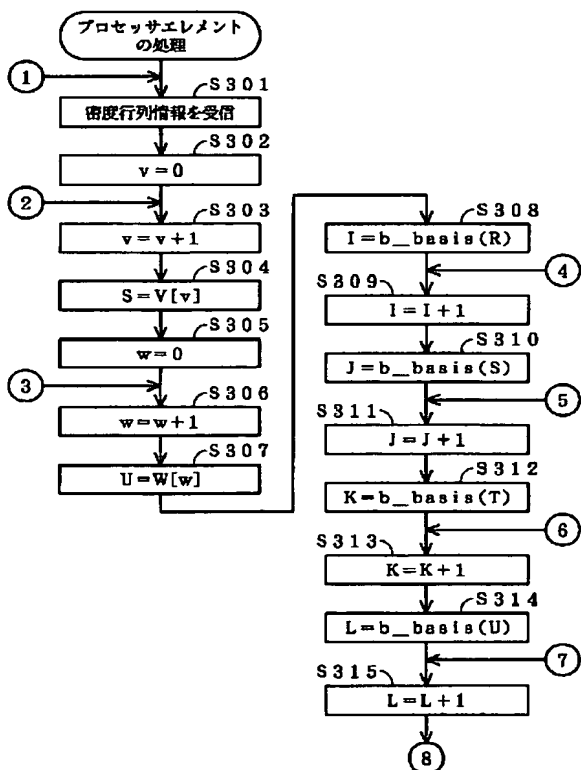
【図 7】



【図 8】

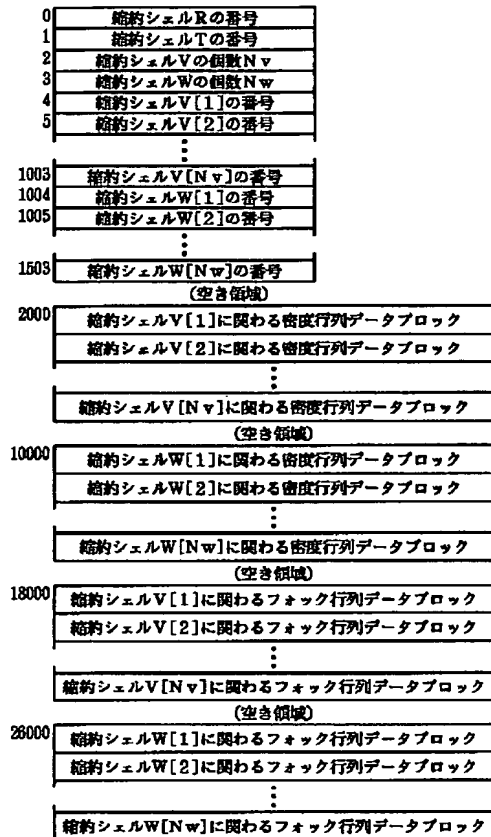


【図 11】

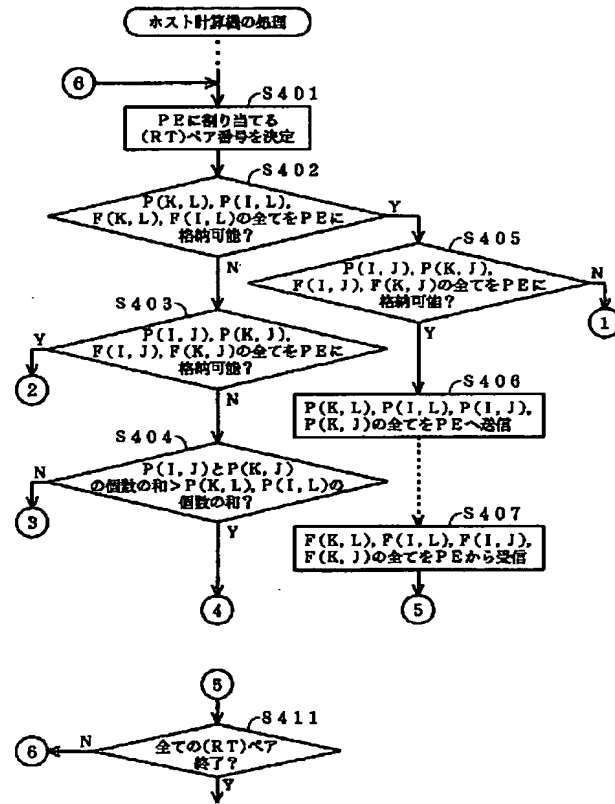


【図 10】

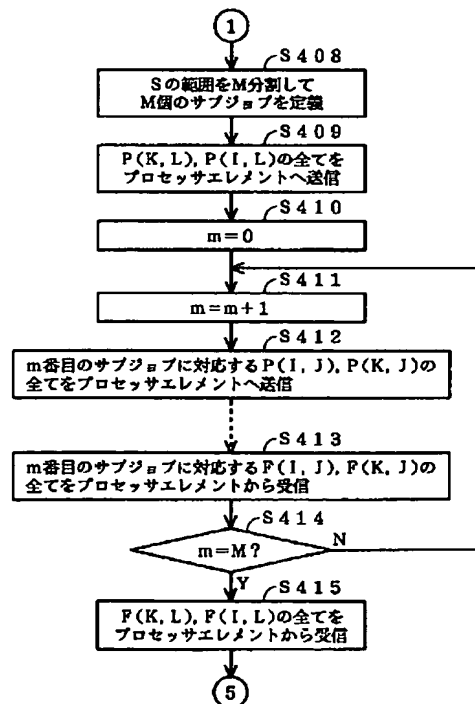
メモリへの行列情報割り付け例



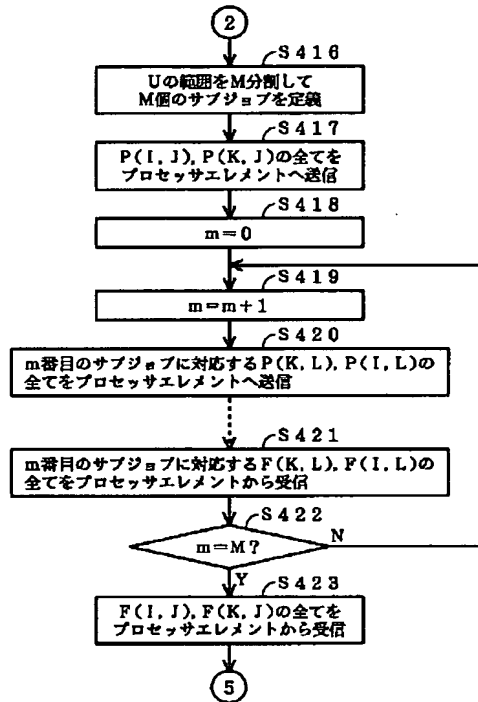
【図 14】



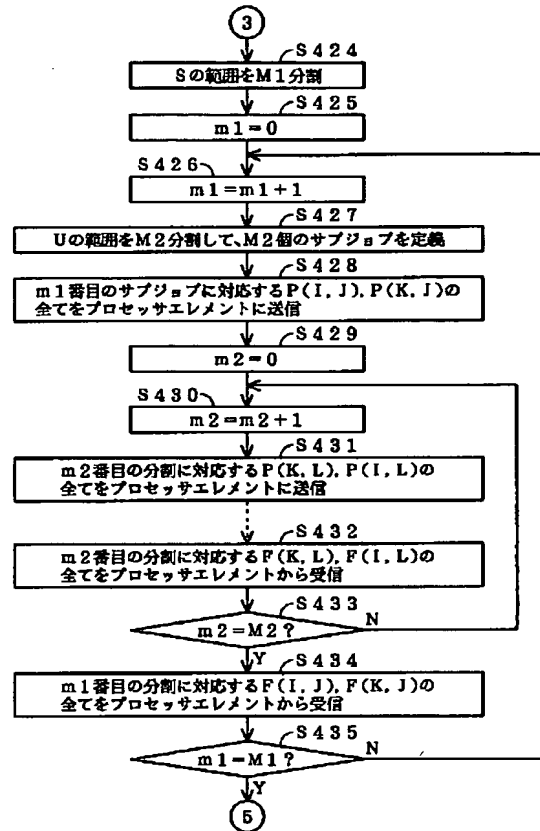
【図 15】



【図 16】



【図 17】

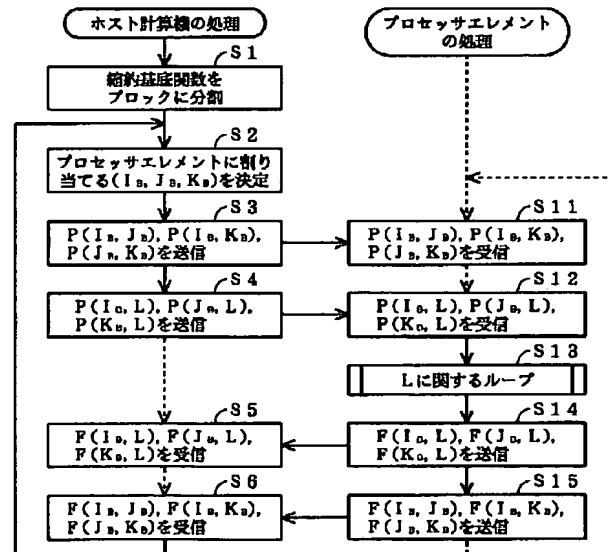


【図 19】

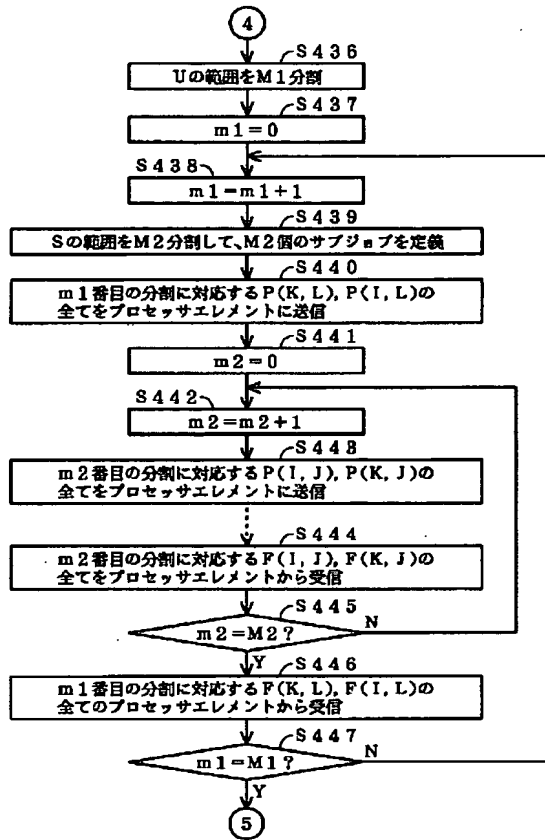
[表1 原始基底関数 i, j, k, l の角運動量、軌道指数、原子核座標]

原始基底関数の番号	角運動量	軌道指数	原子核座標
i	a	ζ_a	A
j	b	ζ_b	B
k	c	ζ_c	C
l	d	ζ_d	D

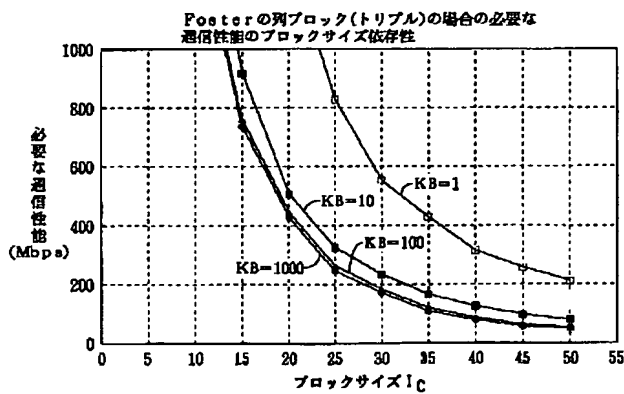
【図 20】



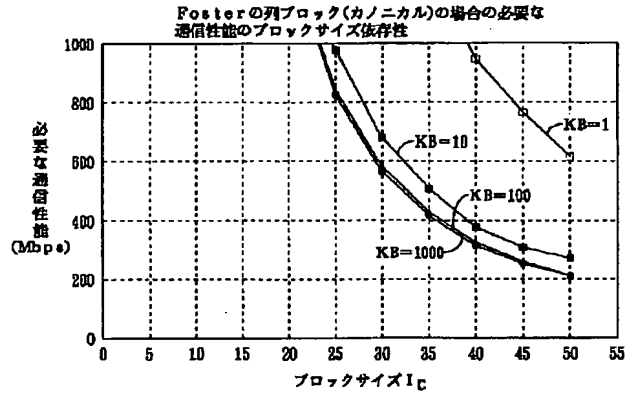
【図18】



【図22】



【図21】



【図23】

(数式1)

$$\phi_{\mu} = \sum_i x_i c_{i\mu}$$

(数式2)

$$\Psi(1, 2, \dots, 2n) = \frac{1}{\sqrt{(2n)!}} \times$$

$$\begin{vmatrix} \phi_1 \alpha(1) & \phi_1 \beta(1) & \phi_2 \alpha(1) & \phi_2 \beta(1) & \dots & \phi_n \alpha(1) & \phi_n \beta(1) \\ \phi_1 \alpha(2) & \phi_1 \beta(2) & \phi_2 \alpha(2) & \phi_2 \beta(2) & \dots & \phi_n \alpha(2) & \phi_n \beta(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_1 \alpha(2n) & \phi_1 \beta(2n) & \phi_2 \alpha(2n) & \phi_2 \beta(2n) & \dots & \phi_n \alpha(2n) & \phi_n \beta(2n) \end{vmatrix}$$

(数式3)

$$H = H_1 + H_2$$

(数式4)

$$H_1 = \sum_p \left[-\frac{1}{2} \nabla_p^2 - \sum_A \frac{Z_A}{r_{pA}} \right]$$

(数式5)

$$H_2 = \sum_p \sum_{q(>p)} \frac{1}{r_{pq}}$$

【図24】

(数式6)

$$\begin{aligned} \varepsilon &= \int \Psi H \Psi d\tau = \int \Psi (H_1 + H_2) \Psi d\tau \\ &= 2 \sum_{\mu} H_{\mu} + \sum_{\mu} \sum_{\nu} (2 J_{\mu\nu} - K_{\mu\nu}) \end{aligned}$$

(数式7)

$$H_{\mu} = \int \phi_{\mu}(1) h^{\text{core}}(1) \phi_{\mu}(1) d\tau_1$$

(数式8)

$$J_{\mu\nu} = \iint \phi_{\mu}(1) \phi_{\nu}(2) \frac{1}{r_{12}} \phi_{\mu}(1) \phi_{\nu}(2) d\tau_1 d\tau_2$$

(数式9)

$$K_{\mu\nu} = \iint \phi_{\mu}(1) \phi_{\nu}(2) \frac{1}{r_{12}} \phi_{\mu}(2) \phi_{\nu}(1) d\tau_1 d\tau_2$$

【図27】

(数式20)

$$\begin{aligned} \chi(r, n, R) &= (r_x - R_x)^{n_x} (r_y - R_y)^{n_y} (r_z - R_z)^{n_z} \\ &\times \sum_m d_m \exp[-\zeta_m (r - R)^2] \end{aligned}$$

(数式21)

$$\begin{aligned} \phi(r, n, R) &= (r_x - R_x)^{n_x} (r_y - R_y)^{n_y} (r_z - R_z)^{n_z} \exp[-\zeta (r - R)^2] \end{aligned}$$

(数式22)

$$\sum_m d_m \exp[-\zeta_m (r - R)^2]$$

(数式23)

$$G(I, J, K, L) = \sum_{m1} \sum_{m2} \sum_{m3} \sum_{m4} d_{m1} d_{m2} d_{m3} d_{m4} g(i, j, k, l)$$

(数式24)

$$g(i, j, k, l) = \iint \phi_i(1) \phi_j(1) \frac{1}{r_{12}} \phi_k(2) \phi_l(2) d\tau_1 d\tau_2$$

(数式25)

$$\begin{aligned} g(i, j, k, l) &= g(i, j, l, k) = g(j, i, k, l) = g(j, i, l, k) = \\ &g(k, l, i, j) = g(l, k, i, j) = g(k, l, j, i) = g(l, k, j, i) \end{aligned}$$

【図 25】

(数式 10)

$$e = 2 \sum_{\mu} \left[\sum_I \sum_J C_{I\mu} C_{J\mu} H_{IJ} \right] \\ + \sum_{\mu} \sum_{\nu} \left\{ \sum_I \sum_J \sum_K \sum_L C_{I\mu} C_{J\mu} C_{K\nu} C_{L\nu} [2G(I, J, K, L) - G(I, K, J, L)] \right\}$$

(数式 11)

$$H_{IJ} = \int \chi_I(1) h^{\text{core}}(1) \chi_J(1) d\tau_1$$

(数式 12)

$$h^{\text{core}}(1) = -\frac{1}{2} \nabla_1^2 - \sum_A \frac{Z_A}{r_{1A}}$$

(数式 13)

$$G(I, J, K, L) = \int \int \chi_I(1) \chi_J(1) \frac{1}{r_{12}} \chi_K(2) \chi_L(2) d\tau_1 d\tau_2$$

【図 29】

(数式 35)

$$[(a+l_1)b, cd]^{(m)} \\ = (P_l - A_l) [ab, cd]^{(m)} + (W_l - P_l) [ab, cd]^{(m+1)} \\ + \frac{N_l(a)}{2\zeta} \left\{ [(a-l_1)b, cd]^{(m)} - \frac{\rho}{\zeta} [(a-l_1)b, cd]^{(m+1)} \right\} \\ + \frac{N_l(b)}{2\zeta} \left\{ [a(b-l_1), cd]^{(m)} - \frac{\rho}{\zeta} [a(b-l_1), cd]^{(m+1)} \right\} \\ + \frac{N_l(c)}{2(\zeta+\eta)} [ab, (c-l_1)d]^{(m+1)} \\ + \frac{N_l(d)}{2(\zeta+\eta)} [ab, c(d-l_1)]^{(m+1)}$$

(数式 36)

$$W = \frac{\zeta P + \eta Q}{\zeta + \eta}$$

(数式 37)

$$\exp = \left[-\frac{\zeta \zeta' (R-R')^2}{\zeta + \zeta'} \right]$$

【図26】

(数式14)

$$\sum_j (F_{IJ} - \varepsilon_{\mu} S_{IJ}) C_{J\mu} = 0$$

(数式15)

$$F_{IJ} = H_{IJ} + \sum_K \sum_L P_{KL} \left[G(I, J, K, L) - \frac{1}{2} G(I, K, J, L) \right]$$

(数式16)

$$\varepsilon_{\mu} = H_{\mu} + \sum_{\nu} (2 J_{\mu\nu} - K_{\mu\nu})$$

(数式17)

$$S_{IJ} = \int \chi_I(l) \chi_J(l) d\tau_l$$

(数式18)

$$P_{KL} = 2 \sum_{\nu} C_{K\nu} C_{L\nu}$$

(数式19)

$$G(I, J, K, L) = G(I, J, L, K) = G(J, I, K, L) = G(J, I, L, K) = \\ G(K, L, I, J) = G(L, K, I, J) = G(K, L, J, I) = G(L, K, J, I)$$

【図30】

(数式38)

$$I \geq J, K \geq L, (IJ) \geq (KL)$$

$$(IJ) = \frac{I(I-1)}{2} + J, (KL) = \frac{K(K-1)}{2} + L$$

(数式39)

$$F_{IJ} = P_{KL} G(I, J, K, L),$$

$$F_{IK} = -\frac{1}{2} P_{JL} G(I, J, K, L),$$

$$F_{IL} = -\frac{1}{2} P_{JK} G(I, J, K, L),$$

$$F_{JK} = -\frac{1}{2} P_{IL} G(I, J, K, L),$$

$$F_{JL} = -\frac{1}{2} P_{IK} G(I, J, K, L),$$

$$F_{KL} = P_{IJ} G(I, J, K, L)$$

【図28】

(数式26)

$$[0_a 0_b, 0_c 0_d]^{(m)} \\ = \frac{1}{\sqrt{\zeta + \eta}} \cdot K(\zeta_a, \zeta_b, A, B) \cdot K(\zeta_c, \zeta_d, C, D) \cdot F_m(T)$$

(数式27)

$$F_m(T) = \int t^{2m} \exp(-Tt^2) dt$$

(数式28)

$$T = \rho(P - Q)^2$$

(数式29)

$$K(\zeta, \zeta', R, R') \\ = \sqrt{2} \frac{\pi^{5/4}}{\zeta + \zeta'} \exp \left[-\frac{\zeta \zeta' (R - R')^2}{\zeta + \zeta'} \right]$$

(数式30)

$$\zeta = \zeta_a + \zeta_b$$

(数式31)

$$\eta = \zeta_c + \zeta_d$$

(数式32)

$$\rho = \frac{\zeta \eta}{\zeta + \eta}$$

(数式33)

$$P = \frac{\zeta_a A + \zeta_b B}{\zeta_a + \zeta_b}$$

(数式34)

$$Q = \frac{\zeta_c C + \zeta_d D}{\zeta_c + \zeta_d}$$

【図31】

(数式40)

$$I \geq J \geq K \geq L$$

(数式41)

$$F_{IJ} = P_{IL} [G(I, J, K, L) + G(I, K, J, L) + G(I, L, J, K)] , \\ F_{IK} = P_{JL} [G(I, J, K, L) + G(I, K, J, L) + G(I, L, J, K)] / 2, \\ F_{IL} = P_{JK} [G(I, J, K, L) + G(I, K, J, L) + G(I, L, J, K)] / 2, \\ F_{JK} = P_{IL} [G(I, J, K, L) + G(I, K, J, L) + G(I, L, J, K)] / 2, \\ F_{JL} = P_{IK} [G(I, J, K, L) + G(I, K, J, L) + G(I, L, J, K)] / 2, \\ F_{KL} = P_{IJ} [G(I, J, K, L) + G(I, K, J, L) + G(I, L, J, K)]$$

(数式42)

$$I_B \geq J_B \geq K_B \geq L_B$$

(数式43)

$$I_B \geq J_B, K_B \geq L_B, (I_B J_B) \geq (K_B L_B)$$

【図32】

(数式44)

$$\frac{2(2I_C^2 + K_B I_C^2) \times 64 \times M}{T_{erl} \times \alpha^2 K_B I_C^4} = \frac{(2 + K_B) \times 128 M}{T_{erl} \alpha^2 K_B I_C^2} \quad (\text{Mbps})$$

(数式45)

$$\frac{2(2I_C^2 + K_B I_C^2) \times 64 \times M}{T_{erl} \times 3 \alpha^2 K_B I_C^4} = \frac{(2 + K_B) \times 128 M}{3 T_{erl} \alpha^2 K_B I_C^2} \quad (\text{Mbps})$$

(数式46)

$$\frac{8 \alpha a^3 R \times 64 \times M}{T_{erl} \times \alpha^2 a^4 R^2} = \frac{512 M}{T_{erl} \alpha a R} \quad (\text{Mbps})$$

(数式47)

$$\exp \left[- \frac{\zeta_A \zeta_B (A-B)^2}{\zeta_A + \zeta_B} \right] < 10^{-15}$$

(数式48)

$$(RT) = \frac{R(R-1)}{2} + T$$

フロントページの続き

(72)発明者 稲畑 深二郎
神奈川県足柄上郡中井町境430 グリーン
テクなかい富士ゼロックス株式会社内

(72)発明者 宮川 宣明
神奈川県足柄上郡中井町境430 グリーン
テクなかい富士ゼロックス株式会社内

(72)発明者 高島 一
東京都豊島区高田3-24-1 大正製薬株
式会社内

(72)発明者 北村 一泰
東京都豊島区高田3-24-1 大正製薬株
式会社内

(72)発明者 小原 繁
北海道釧路市武佐3-12-358

Fターム(参考) 5B045 AA07 GG12 KK04
5B049 BB07 EE03 EE04 EE41 GG07